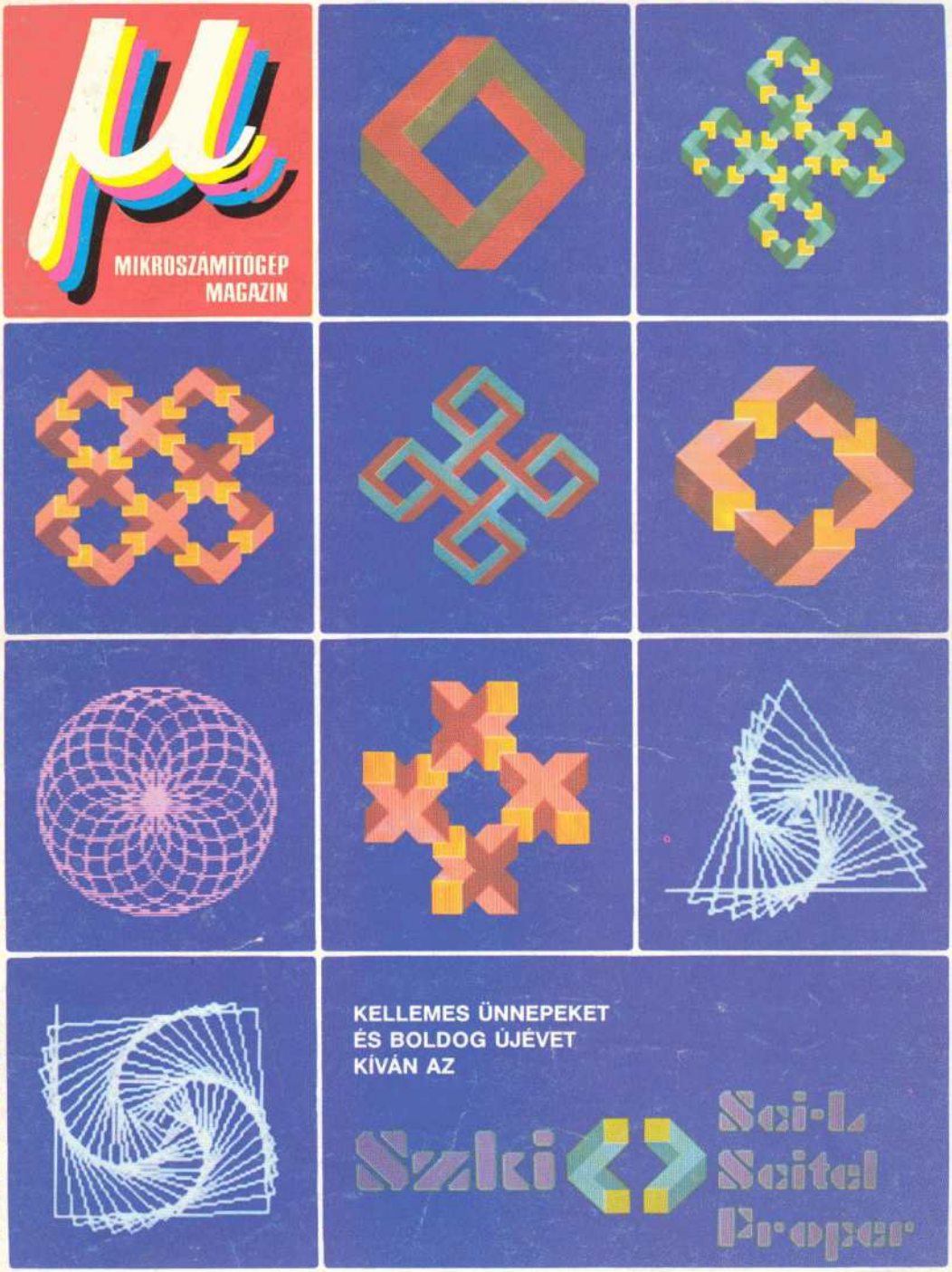
SZÁZOLDALAS * SZÁZOLDALAS * SZÁZOLDALAS



SZÁZOLDALAS * SZÁZOLDALAS * SZÁZOLDALAS * SZÁZOLDALAS

1986 11–12. Ára 60,– Ft

* SZÁZOLDALAS * SZÁZOLDALAS *

AZ ALKOTÓ IFJÚSÁG EGYESÜLÉS SZÁMÍTÁSTECHNIKAI IRODÁJA

A COMMODORE 64 típusú számítógéppel rendelkező vállalatok figyelmébe ajánlja

— a mérnöki-tervezői munka könnyítésére

épületgépészeti
kivitelezői költségvetési
fűtésirendszer-csőméretezési
fűtőtesthőveszteség-számítási
kéményméretezési
padlófűtés-méretezési
tüzeléstechnikai
légcsatorna-méretezési
porleválasztó-méretezési
gőzvezeték-méretezési
statikai
geodéziai
gépipari tervezői
hálózatitrafó-méretezési

programcsomagjait

— az irodai-ügyviteli feladatok könnyítésére

létesítményjegyzék
iparstatisztikai
mérlegellenőrző
pénzügyi
forgóeszköz-nyilvántartó
késedelmi kamat
menetlevél-feldolgozó
fuvarellenőrző
munkaügyi
vgm- és gmk-elszámoló
társadalombiztosítási
szabadság-nyilvántartó
bérelemző
érdekeltségialap-elszámoló
keresetszabályozó
keresetiadó-számító

programcsomagjait

valamint

— a közismert TRACE—64, DBASIC V.1.0, DBASIC V.1.1, EXPANDER, ékezetes karakterkészlet- és grafikai segédprogramokat

További részletes felvilágosítás:

ALKOTÓ IFJÚSÁG EGYESÜLÉS
1066 BUDAPEST VI., JÓKAI U. 8.
TELEFON: 124-479, 314-121. TELEX: 22-7272
LEVÉLCÍM: 1519 BUDAPEST, PF. 330.

MIKROSZÁMÍTÓGÉP MAGAZIN

4. ÉVFOLYAM
1986. 11–12. SZÁM

A NEUMANN JÁNOS SZÁMÍTÓGÉP-TUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

E számunkat szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Lindner László
(sakkprogramozás)

Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Vadkerti János
(μprogramok)

Varga András
(iskola-számítógép)

Felelős szerkesztő:
Könyves Tóth Pál
Szerkesztőség:
1027 Budapest II., Fő u. 68.
Telefon: 154-250
Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat
Felelős kiadó:
dr. Petrus György igazgató
Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660
Terjeszti a Magyar Posta Előfizethető a hírlapkézbesítő hivataloknál és a Posta Hírlapelőfizetési és Lapellátási Irodáján (1900 Budapest V., József nádor tér 1.) vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.
Egy szám ára 30,— Ft
Előfizetési díj:
fél évre 180,— Ft
egy évre 360,— Ft
Külföldön terjeszti a Kultúra, 1389 Budapest, pf. 149. és a Magyar Média 1932 Budapest, pf. 279. 86—253

Szikra Lapnyomda
Budapest (86-4316)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Címképünk:

Domán András és F. Farkas Tamás Proper 16 W gépen készült grafikái

Tartalom

Számadás	3
A minőségügy — közügy	20
Okos kártyák	42
Olvastunk . . .	44
Barátunk, a számítógép	46
Hasznos tanácsok floppy-tulajdonosoknak	47
Helyi hálózatok	50
Számítógépes adatbankok mindenkinek	56
Adok-veszek-cserélek	68
Pagodák között	69
Hardver, szoftver — vagy valami más?	76
Az egér és a mozgatógömb	84
Zeneszerzés számítógéppel	90
Azt írja az újság . . .	94
Távoktatás és informatika	96
ISKOLA — SZÁMÍTÓGÉP	
Az ismeretlen C16	4
Turbo Tape	5
A képernyőtartalom kimentése	6
Oktatóprogram	7
Hangszerprogram HT géphez FORTH nyelven	8
Pontsor közelítése polinommal	9
Függvénybemenet	10
Játsszunk számítógépet!	11
Milyen a jó számítógépes játék?	12
Számítógép a biológia oktatásában	13
Beszámoló a SZOFIT '86 programozói versenyéről	13
Nyomkövető program	14
DIÁKROVAT	
Öninduló program	22
Átszámozó program	22
A Spectrum rendszerváltozói	23
Softcopy CBM MPS802-re	24
PROGRAMOZÁSTECHNIKA	
FORTH rekordszerkezetek	25
Z80 programozási gyakorlatok 5.	26
Az UNIX operációs rendszer	29
Szorzás és osztás assembly programokban	35
Verem	37
Adatbázis-kezelés	38
Tömörítő rutin	40
Folttervező	41
BASIC és gépi kód	42
μPROGRAMOK	
List-Microsoft BASIC-nél	59
Adatbevitel-egyszerűsítő	59
Egy menüprogram	60
Cirill betűk	62

Néhány képernyőkezelő rutin 63
A Beta basic és alkalmazása 63
A Melbourne Draw 65
Programgyorsító rutinok 66
256 × 192-es grafika 67
A képernyő invertálása 67
Rajzoló 68

µKLUB

A lemezmeghajtó egységszámának beállítása 70
A lemezkapacitás növelése 72
Szövegkereső 73
Amerikai előzetes 73
Sztringkereső 74
Eredeti karakterek négyszeres nagyításban 75
Kör és egyenes rajzolása 76

SAKKPROGRAMOZÁS

Bitek és figurák 77
Mephistók a világ élvonalában 78

JÁTÉKPROGRAMOK

Ha nehéz a Quasimodo (avagy Hanch Back) 80
Rázós úton 80
Játékprogram-módosítás 80
Coby-vadászat 81
A megoldott Tantalizer 82

PIAC

Nyári árak Bécsben 86

AZ OLVASÓ ÍRJA 87

KÖNYVEK 92

HÍREK, ÉRDEKESSÉGEK 93

PÁLYÁZATI FELHÍVÁS

Az NJSZT, a szekszárdi Garay János Gimnázium és a Mikroszámítógép Magazin Szerkesztősége ismét pályázatot hirdet az alábbi kategóriákban:

1. Új, önálló játékprogram készítése HT—1080Z, Sinclair Spectrum, Commodore 16, Commodore Plus/4 és 64 gépre.
2. Valamely tantárgyhoz kapcsolódó, a tanítási órát segítő, illetve a tanulók önálló tanulását támogató oktatóprogram készítése a fent említett gépekhez.

A pályázaton részt vehet minden általános és középiskolás tanuló, valamint elsőéves egyetemista.

A pályázat beadása: II. 15-ig a programot mágnesszalag-kazettán (a kazettára többször felvéve), vagy mágneslemezen (floppy) rövid leírás kíséretében (mit tud, hogyan működik a játék, hányadik osztály mely tantárgyának melyik anyagrészéhez kapcsolódik stb.), jeligével ellátva (külön zárt borítékban a név, lakcím vagy iskola) kérjük beküldeni a Mikroszámítógép Magazin Szerkesztőségébe (Budapest, Fő u. 68. 1027). A szerkesztőségnek joga van a pályázaton részt vett programok közlésére, amiért a szokásos honoráriumot fizeti. A döntő 10-10 résztvevőjét az NJSZT tagjaiból, a Garay Gimnázium tanáraiból és a szerkesztőség munkatársaiból álló előzsűri választja ki.

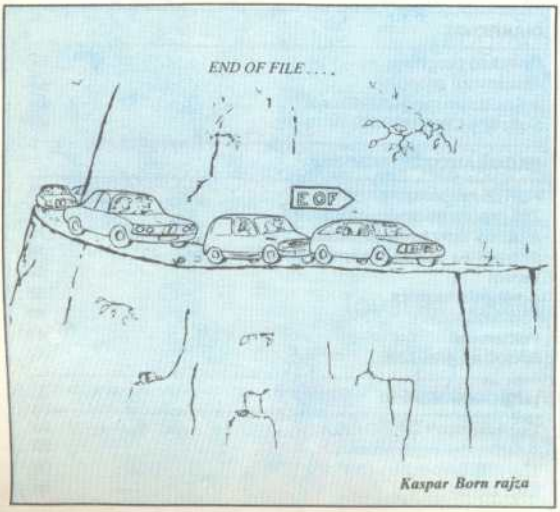
A döntő — melyen az előzsűri által kiválasztott 10-10 program versenyez — Szekszárdon, a Garay János Gimnáziumban rendezendő Garay napok alkalmából, 1987. márciusában lesz.

A döntőbe jutott tanulókat a Garay János Gimnázium vendégül látja.

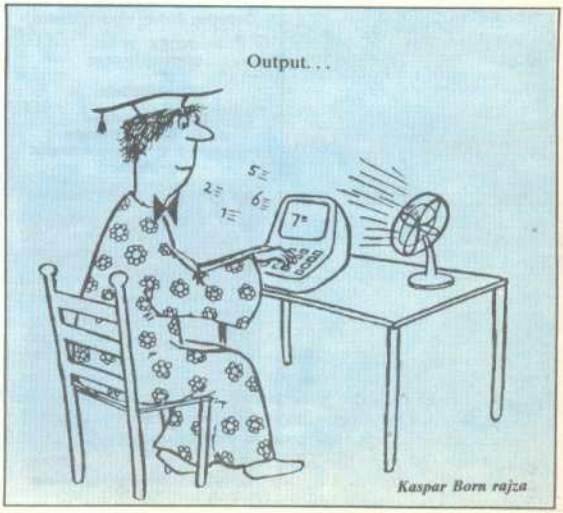
Mindkét kategóriában az első három helyezett programot díjazzuk, és mindkét kategóriában kiadjuk a közönség díját.

NJSZT	Garay János	Mikroszámítógép
Dömölki Bálint	Ált. Gimn.	Magazin Szerk.
elnök	Zentai András	Kovács Győző
	igazgató	a szerk. biz. vezetője

Minden kedves Olvasónak kellemes ünnepeket és boldog új évet kíván a Szerkesztőség!



Kaspar Born rajza



Kaspar Born rajza

Számadás

„Úgy látom, igazán hasznos és szép anyagot nem találok, mert az előttem járók már minden fontosat, hasznosat magukhoz ragadtak, úgy teszek hát, mint az egyszeri szegény ember, aki utolsónak ért a vásárba, és mivel más nem maradt neki, összezedte a silány holmit, ami a sok vásárló közül egyiknek se kellett. Én is ezt a maradékot rakom majd gyenge kis öszvérem hátára. Nem a nagyvárosokban kínálom a portékámat, hanem a szegény falvakban, és megelégszem olyan árral, amilyet a portékám ér."

(Leonardo da Vinci)

A μM ez évi számait lapozgatom, olvasói leveleket böngészek, hiszen ebben az évben ez az utolsó alkalom, hogy egész éves munkánkról mérleget készítsünk, és jövő évi terveinkről egy kicsit elbeszélgessünk.

Amikor 1983-ban a lapot alapítottuk, egy olyan magazint képzeltünk el, ami az informatika fontos kérdéseivel népszerű módon foglalkozik, ugyanakkor támogatjuk az iskolaszámítógép-programot, a számítógépes klubmozgalmat, a számítógép-építő amatőröket. Az írások szerzőit arra biztattuk, hogy mondanivalójukat mindenki számára érthetően fogalmazzák meg, nem jelentettünk meg olyan cikkeket, amelyeket olvasóink többsége — véleményünk szerint — nem, vagy csak nehezen tudott volna megérteni.

Arra törekedtünk, hogy a lapban megjelent írások szoros kapcsolatban legyenek a gyakorlattal, a túlzottan elméleti eszmefuttatásokat átengedtük a μM-nél tudományosabb igényű sajtótermékeknek.

A magunk módján politizáltunk is, foglalkoztunk olyan gazdasági kérdésekkel, amelyek nem csak a szakembereket, de az informatikával foglalkozókat is érdekelték.

Megpróbáltuk a lapot az általunk elképzelt szinvonalon tartani, olvasóinknak kell eldönteni, hogy ezt a célt mennyire sikerült elérnünk.

A lap első számait forgatva talán az első pillanatban fel sem tűnik, hogy az utóbbi időben a μM szerkezete némileg megváltozott, úgy véljük, hasonló módon, mint ahogyan az informatika és a magyar társadalom viszonya is változott. Egyre több jelzést kaptunk — már a múlt évben is —, hogy olvasóink körében csökken az érdeklődés a nagyon kezdőknek szóló és túl általánosan fogalmazó írások iránt, elsősorban a diákok, de a nem számítástechnikus és az informatika iránt érdeklődő „nagyközönség" körében is. Egyre többen javasolták, hogy csökkentsük a játékprogramok mennyiségét, helyette inkább programozói fogásokat, illetve olyan programokat közöljünk, amelyeket a napi munkában, az iskolában, a háztartásban lehet alkalmazni. Olvasóink hasznos tanácsokat, ötleteket vártak és nem kész megoldásokat, amelyeket — egy levélből idézek — „szolgai módon be kell billentyűzni a gépbe, és így az embert megfosztja az alkotás örömétől."

Ezért aztán tartalmilag néhány rovatunk átalakult (pl. az Iskola-számítógép, a Sakkprogramozás és a μKlub), új rovatokat hoztunk létre (μprogramok, Programozástechnika), és természetesen folyamatosan keressük azokat a lehetőségeket, amelyeknek a bevezetésével a μM-t még érdekesebbé és tartalmasabbá tehetjük.

Az Iskola-számítógép rovat egyre inkább a tanárok, de a diákok szakmai, metodikai és talán még pedagógiai fórumává is vált. Az elmúlt évben egyre több olyan írás jelent meg, amelyben pedagógusok mutatják be oktatási programjaikat, vagyis azt, hogy a számítógépeket hogyan alkalmazzák a tanórákon. Nagyon sajnálatos tapasztalat, ami talán a jelenlegi helyzetet is jellemzi, hogy alig jelent meg olyan írás, amelyben egy történelem- vagy irodalomtanár mutatna be oktatóprogramokat, amelyeket az órákon is rendszeresen használ. „Sajnos" még ma is az a helyzet, hogy az iskolai számítógépeket inkább a technika-, a fizika- és a matematikatanárok használják leginkább.

Nagyon nem szeretnénk, ha a lap kizárólag a programozóknak szólna és esetleg azt a sokszor hallott hamis formulát idézné, hogy az informatika azonos a programozással. Ezért a programleírásokat igyekszünk úgy elkészíttetni, hogy az a feladat előkészítéséről, a megoldásról, az algoritmus kiválasztásáról is hasznos információkat adjon. Ezt a célt szolgálta a játékprogramozás technikájáról szóló sorozatunk is, ami — reméljük — hasznos ismereteket adott más programok írásához is.

Nagyon nehezen tudjuk a diákrovat folytonosságát biztosítani, amit pedig azért hoztunk létre, hogy lehetőséget adjunk a kísérletező kedvű ifjúságnak eredményeik bemutatásához. A vakáció után általában kisebb-nagyobb nehézségekkel kell megküzdenünk, hogy a diákszerkesztőség munkája ismét megindúljon.

Befejeztük az „Építsünk számítógépet" sorozatot, egyelőre nincs szándékunk egy újabb gép építéséről sorozatot kezdeni. Sokszori erőfeszítésünk ellenére nem sikerült a hazai gyártókat számítógép-építő kit eladására ösztökélni, nem láttak üzletet egy ilyen akció megindításában. Sajnos az első sorozatnak „hibája" is az volt, hogy — különösen vidéken — az amatőrök nem tudták a szükséges alkatrészeket beszerezni, így a gépek épültek, de befejezni a munkát csak nagyon kevesen voltak képesek. Ezért egyelőre a hardver amatőrök részére olyan írásokat adunk közre, amelyek különleges perifériák, meglévő gépekhez hardverkiegészítések, interfészek építéséhez adnak segítséget. Ha közben valamelyik gyártó mégis meggondolja magát és elkezd számítógép kit-et gyártani, akkor örömmel adunk helyet egy újabb számítógép-építő sorozatnak is.

Ebben az évben az első számunk februárban került az olvasókhoz, összesen tizenegyszer jelentünk meg. Olvasóinkat mégsem csaptuk be, hiszen ez az év végi dupla szám teljessé teszi a sorozatot. A havilappá válással sikerült a megjelenésünket pontossá tenni, minden hónap első munkanapján a lap kapható volt az újságárusoknál, és néhány szám kivételével mindig el is fogyott.

Sajnos — mind ez ideig — az előkészítési munka idejét három hónapnál rövidebb időre nem sikerült leszoritanunk, ami azt jelenti, hogy még ma sem tudunk friss és aktuális cikkeket közölni, például a számítógépárakról. Mire piaci tanácsadónk megjelenne, addigra az árak olyan nagyot változnak, hogy a cikk információértéke enyhén szólva is vitatható. Most ismét azt reméljük, hogy a következő évben ezt a problémát is meg tudjuk oldani, különösen akkor, ha sikerül a szerkesztőségi munkát bizonyos mértékben automatizálni, és a cikkeket a szerkesztőségből mágneses adathordozón a nyomdának átadni.

Ez év márciusában az Idegenforgalmi és Propaganda Vállalattal együttműködve megrendeztük — μ'86 néven — az Első országos mikroszámítógépes találkozót. Ez az esemény amellett, hogy bemutatta a hazai mikroszámítógép-tervezés, -gyártás és -alkalmazás eredményeit, a hazai amatőrizmus és klubmozgalom fejlődését, jó alkalmat adott a számítástechnikai lapok és az olvasók találkozására is. Már folynak az előkészületek a μ'87 megrendezésére, amelyben a μM ismét komoly szerepet fog vállalni.

A lap életében — reméljük — meghatározó esemény volt a KISZ KB és az NJSzT közötti szerződés aláírása, és így a μM a KISZ KB lapjává is vált. Ez az együttműködési megállapodás azt jelenti, hogy a KISZ és az NJSzT az informatika társadalmi elterjesztésében közös és egyeztetett úton halad, a μM-t is ezen cél megvalósításának szolgálatába állítja. Azt reméljük, hogy ezzel az együttműködéssel nemcsak a lap olvasótábora, de a hatása és a befolyása az ifjúság körében is növekedik. Nem kevésbé fontos eredménye lehet ennek az együttműködésnek, hogy új ötleteket és javaslatokat kaphatunk az ifjúsági szervezet tagjaitól, a KISZ külföldi kapcsolatai révén pedig — elsősorban a szocialista országokban — a lap nemzetközi szerepe erősödhet lényegesen.

A KISZ-szel való együttműködés egyben a példányszám további növekedésének és — a nagyobb kapcsolatrendszer eredményeképpen — a tartalmi gazdagodásnak is a biztosítéka.

KOVÁCS GYÖZŐ

Az ismeretlen C16

KERNAL RUTINOK 3.

A beviteli/kiviteli KERNAL rutinok újabb összefüggő csoportját, a soros busz használatával kapcsolatos rutinokat ismertetem. A korábbiaktól eltérően ezekről személyes tapasztalataim alig vannak; a rájuk vonatkozó információkat a rendelkezésemre álló forrásmunkákban található, gyakran egymásnak ellentmondó adatokból gyűjtöttem össze.

A korábban ismertetett beviteli/kiviteli rutinok közvetlenül hívják ezeket a rutinokat, ha az aktuális fizikai egységszám 3-nál nagyobb. Így a most leírtak használatát könnyen nélkülözhetjük, csak a teljesség kedvéért írok róluk. Akit részletesebben érdekel a téma, bőséges irodalom áll rendelkezésére magyar nyelven is.

Az itt leírt összes rutin azonos módon működik a C—64-en és a VC—20-on is, a hivatkozott tárcímek is azonosak. A végrehajtás hibátlan voltáról az I/0 állapotszó ($90) lekérdezésével győződhetünk meg.

Az eszköz általában nyomtató vagy mágneslemezegység. Angol nyelvű leírásban „DMA disk"-re történő hivatkozást is találtam, nem tudom, hogy ez azonos-e a legutóbbi részben említett „Rendszerváltozók és I/0 címek" című füzet 31. oldalán szereplő „Kennedy-eszköz"-zel.

Még röviden a szóhasználatról: a „másodlagos cím" kifejezést használom a megnyitási mód, illetve csatornaszám helyett is. A soros buszra felfüzött és bekapcsolt eszközök alaphelyzetben *figyelő* állapotban vannak. A *fogadó* állapotban lévő eszköz kész a soros buszon kapott adatok fogadására, feldolgozására. *Hallgató* állapotúnak is nevezik. A *beszélő* állapot az, amikor az eszköz kész adatok küldésére.

Ezután vegyük sorra a rutinokat!

SECOND $FF93 65427

A másodlagos címet küldi el a fogadó állapotú eszköznek. A másodlagos címet a rutin hívása előtt az A regiszterbe kell tölteni. Az eszközt előzőleg a LISTEN rutinnal kell fogadó állapotúvá tenni.

TKSA $FF96 65430

A másodlagos címet küldi el a TALK rutinnal beszélő állapotúvá tett eszköznek. Hívása előtt a másodlagos címet az A regiszterbe kell tölteni.

ACPTR $FFA5 65445

A soros buszon a beszélő eszköztől egy adatbájtot fogad, mely az A regiszterbe kerül. Az eszközt a TALK rutinnal kell előzetesen beszélő állapotba hozni.

CIOUT $FFA8 65448

A LISTEN rutinnal fogadó állapotba hozott eszköz részére egy adatbájtot küld a soros buszon keresztül. A küldendő bájtot az A regiszterbe kell a rutin hívása előtt tölteni. A kivitel késleltetve, pufferen keresztül történik: minden hívásnál az előző híváskor küldött adat kerül a soros buszra, az utolsó bájtot az UNLSN hívásával küldjük el. A puffer a $95 címen van, a $94 címen egy jelző (flag) van, melynek tartalma $00, ha a puffer üres, és $80, ha a pufferben adat van.

UNTLK $FFAB 65451

A beszélő állapotú eszközt figyelő állapotúvá teszi. Hatására a számítógép felé irányuló adatáramlás megszakad.

UNLSN $FFAE 65454

Az összes fogadó állapotú eszközt figyelő állapotba hozza.

LISTEN $FFB1 65457

A figyelő beviteli/kiviteli eszközt fogadó állapotba hozza, azaz felszólítja a soros buszon át küldött adatok fogadására. A hívás előtt az eszköz számát, amely megegyezik a BASIC-ben is használatos eszközszámmal, az A regiszterbe kell tölteni.

TALK $FFB4 65460

A figyelő állapotú beviteli/kiviteli eszközt beszélő állapotba hozza, vagyis adatküldésre szólítja fel a soros buszon keresztül. Hívása előtt az egységszámot az A regiszterbe kell tölteni.

BARNA LÁSZLÓ

Oktatási programpályázat az NSZK-ban

A CHIP számítógép-magazin augusztusi száma szerkesztőségi cikkben számol be arról, hogy milyen eredményei vannak az NSZK-ban a múlt évben első alkalommal kiírt oktatási szoftverpályázatnak. A pályázat elsődleges célja — írja a cikk —, hogy feltérképezzék a kereskedelemben kapható szoftvertermékek közül az iskolák számítógépein használható programokat. Ezenkívül rá kívánták irányítani a figyelmet az új, minőségileg magas színvonalon álló programokra és azok szerzőire is.

Ennek megfelelően eleve két kategóriában történt az értékelés: a már kereskedelmi forgalomban megvásárolható programokra, valamint a még nem forgalmazott új fejlesztésekre. A zsűri mindkét kategóriát további 20-20 részterületre osztotta.

Oktatási témák szerint
nyelvoktatás,
matematika és a természettudományok,
művészetek,
informatika,
műszaki és gazdasági oktatás.

Iskolatípusok és -fokozatok szerint
népoktatás (ált. iskola alsó tagozat),
általános iskola felső tagozat I.,
általános iskola felső tagozat II.,
szakoktatás.

A kiírás alapkövetelménye szerint a pályázóknak figyelembe kellett venniük az érvényes tanterveket, és — a számítógépek kínálta lehetőségek felhasználásával — használható, új módszertani elemeket kellett bemutatniuk. További követelmény volt, hogy a pályázatra nevezett program ne helyettesítse, csupán ösztönözze a pedagógust az új lehetőségek kiaknázására.

A beérkezett közel 100 pályamunka közül a magasan kvalifikált tagokból álló zsűri sok programról azt állapította meg, hogy a fiatalok számára megrendezett „Arany Mágneslemez" programozói verseny színvonalát sem éri el. Kissé ironikusan fogalmazzák meg a kiírás általános tapasztalatát: az oktatásban használható programok tekintetében az NSZK a szó valódi értelmében a fejlődő országokhoz sorolható.

Végeredményben mindössze két kategóriában adtak ki díjat, továbbá két különdíjat ítéltek oda. A díjazott pályázatok közül öt matematikai-természettudományi, kettő nyelvi, egy-egy pedig műszaki és társadalomtudományi témát dolgozott fel. A programok túlnyomó többsége Apple II. típusú számítógépen fut.

A zsűri értékelése szerint a kevés kiadott díj ellenére az elfogadott pályamunkák magas minőségi igényeknek tesznek eleget. Remélik, hogy a pályázat meghirdetésével újabb lendületet adtak az oktatás területén használható programok fejlesztéséhez, és az oktatásban az egyre nagyobb teljesítményű és korszerűbb gépek mellett mihamarabb komplett szoftverenyagok is rendelkezésre fognak állni.

A CHIP cikke alapján összeállította:
DR. BALLA LÁSZLÓ

C–16

TURBO TAPE

A C—16-nak sok jó tulajdonsága mellett néhány bosszantó hátránya is van. Ilyen például a magnókezelés kibírhatatlan lassúsága. Egy komolyabb játék betöltése magnóról körülbelül nyolc percig tart. Ezenkívül a C—16 magnóformátuma nem kompatibilis a C—64-ével, így a C—64 programok a C—16-ba magnóról nem tölthetők be.

Ezeken a problémákon segít az alábbi program. Az adatátvitel sebességét mintegy tizenötszörösére növeli (500 bájt/s), a megbízhatóság romlása nélkül. A fájl formátuma megegyezik a C—64 TURBO TAPE-formátummal; így a C—16-programok másolására használhatók a C—64-en rendelkezésre álló turbo-másolók, például a Copy 190. A program a BASIC RAM-ból egyetlen bájt helyet sem foglal el.

Ez a program az általam írt gépi kódú program DATA-sorokba átírt változata. Begépelés után, mielőtt futtatnánk, feltétlenül mentsük ki, mert futás közben önmagára írja a TURBO TAPE 16-ot! Futtatás után előáll a kész program, amely normál BASIC-programként szalagra vehető.

A TURBO TAPE 16 RUN paranccsal indul, bemásolja magát a szabad helyekre, majd a BASIC-területet felszabaditja. Az eredeti LOAD, SAVE, VERIFY utasítások továbbra is használhatók. Az új utasítások:

← s:TURBO SAVE
← l:TURBO LOAD
← v:TURBO VERIFY

RÉTVÁRI GYÖRGY

Szerkesztői megjegyzés:
A Supertape elnevezésű, gépfüggetlen, kazettaformátum C—16 használata esetén is 3600—7200 baud átviteli sebességet tesz lehetővé, és — mert gépfüggetlen — lehetővé teszi gyakorlatilag minden ismert gép között kazettán az adatátvitelt. Néhány gépre a HCC-tagok számára díjmentesen rendelkezésre áll az író/olvasó program.

```
10 DATA 1,16,63,19
20 DATA 39,16,194,7,158,52,49,51
30 DATA 55,58,143,34,20,20,20,20
40 DATA 20,20,20,20,20,20,20,20
50 DATA 84,85,82,66,79,32,84,0,5
60 DATA 80,69,32,49,54,0,0,0
70 DATA 162,108,189,211,18,157,95,2
80 DATA 202,208,247,162,111,189,100,18
90 DATA 157,255,3,202,208,247,189,225
100 DATA 16,157,0,6,202,208,247,162
110 DATA 132,189,224,17,157,255,6,202
120 DATA 208,247,169,0,162,6,141,2
130 DATA 3,142,3,3,169,31,162,6
140 DATA 141,8,3,142,9,3,32,79
150 DATA 255,147,13,32,32,67,45,54
160 DATA 52,32,67,79,77,80,65,84
170 DATA 73,66,76,69,32,84,85,82
180 DATA 66,79,32,84,65,80,69,32
190 DATA 70,79,82,32,67,45,49,54
200 DATA 13,13,32,32,32,32,32,32
210 DATA 32,87,82,73,84,84,69,78
220 DATA 32,66,89,32,71,89,79,82
230 DATA 71,89,32,82,69,84,86,65
240 DATA 82,73,13,13,13,32,32,32
250 DATA 32,83,65,86,69,58,95,83
260 DATA 32,32,32,32,76,79,65,68
270 DATA 58,95,76,32,32,32,32,86
280 DATA 69,82,73,70,89,58,95,86
290 DATA 13,13,0,104,104,76,123,138
300 DATA 162,255,134,58,32,90,136,134
310 DATA 59,132,60,32,115,4,170,240
320 DATA 239,176,3,76,46,135,32,83
330 DATA 137,32,121,4,76,34,6,32
340 DATA 115,4,240,4,201,95,240,3
350 DATA 76,217,139,32,115,4,201,76
360 DATA 208,3,76,0,4,201,86,208
370 DATA 3,76,3,4,201,83,240,3
380 DATA 76,161,148,32,115,4,32,107
390 DATA 168,162,3,181,43,149,178,202
400 DATA 16,249,32,25,227,144,4,8
410 DATA 76,204,140,32,40,242,32,96
420 DATA 2,169,5,32,207,6,165,173
430 DATA 24,105,1,202,32,241,6,162
440 DATA 23,185,178,0,32,241,6,162
450 DATA 21,200,192,5,208,243,160,0
460 DATA 162,19,177,175,196,171,144,3
470 DATA 169,32,202,32,241,6,162,20
480 DATA 200,192,187,208,237,169,2,32
490 DATA 207,6,152,32,241,6,132,245
500 DATA 162,22,177,178,32,241,6,162
510 DATA 18,230,178,208,4,230,179,202
520 DATA 202,165,178,197,180,165,179,229
530 DATA 181,144,231,165,245,32,241,6
540 DATA 162,22,136,208,246,169,0,133
550 DATA 144,32,200,232,76,220,139,133
560 DATA 144,160,0,169,2,32,241,6
570 DATA 162,22,136,192,9,208,244,162
580 DATA 20,198,144,208,238,152,32,241
590 DATA 6,162,22,136,208,247,202,202
600 DATA 96,133,182,69,245,133,245,169
610 DATA 8,133,183,6,182,165,1,9
620 DATA 2,32,18,7,162,32,41,253
630 DATA 32,18,7,162,29,198,183,208
640 DATA 234,96,202,208,253,144,6,162
650 DATA 18,202,208,253,234,133,1,96
660 DATA 132,144,32,137,241,173,53,3
670 DATA 56,237,51,3,8,24,101,178
680 DATA 133,180,173,54,3,101,179,40
690 DATA 237,52,3,133,181,32,121,2
700 DATA 132,245,32,158,2,196,147,208
710 DATA 2,145,178,209,178,240,4,5
720 DATA 144,133,144,69,245,133,245,230
730 DATA 178,208,2,230,179,165,178,197
740 DATA 180,165,179,229,181,144,219,32
750 DATA 158,2,32,200,232,165,182,69
760 DATA 245,5,144,240,4,169,255,133
770 DATA 144,24,166,180,164,181,32,10
780 DATA 168,76,220,139,169,0,44,169
790 DATA 1,133,10,133,147,32,115,4
800 DATA 32,107,168,164,43,165,44,132
810 DATA 178,133,179,32,112,2,201,0
820 DATA 240,249,133,183,32,158,2,153
830 DATA 51,3,200,192,192,208,245,32
840 DATA 200,232,165,183,201,2,240,8
850 DATA 201,1,208,223,165,173,240,10
860 DATA 173,51,3,133,178,173,52,3
870 DATA 133,179,169,52,133,182,169,3
880 DATA 133,183,32,232,233,144,4,8
890 DATA 76,204,140,164,171,240,17,169
900 DATA 175,141,223,7,136,32,217,7
910 DATA 217,56,3,208,174,152,208,244
920 DATA 76,32,7,32,141,227,76,100
930 DATA 227,32,27,227,144,4,8,76
940 DATA 204,140,96,32,102,2,32,96
950 DATA 241,76,124,2,32,102,2,32
960 DATA 96,2,32,171,2,38,182,165
970 DATA 182,201,2,208,245,160,9,32
980 DATA 158,2,201,2,240,249,196,182
990 DATA 208,232,32,158,2,136,208,246
1000 DATA 96,162,8,32,171,2,38,182
1010 DATA 202,208,248,165,182,96,169,16
1020 DATA 36,1,240,252,36,1,208,252
1030 DATA 45,9,255,72,169,224,141,2
1040 DATA 255,169,0,141,3,255,169,16
1050 DATA 141,9,255,104,105,254,96
1060 DATA 124
63000 READ K:READ A:K=K+256*A
63010 READ V:READ A:V=V+256*A
63020 FOR I=K TO V
63030 READ A:POKE I,A:O=(O+A) AND 255
63040 NEXT
63050 READ A:IF A=0 THEN 63070
63060 PRINT "HIBA AZ ADATOKBAN !":STOP
63070 POKE 43,K AND 255:POKE 44,K/256
63080 A(0)=V+1:POKE 45,A(0) AND 255
63090 POKE 46,A(0)/256
```

COMMODORE 16

A képernyőtartalom kimentése

Az alábbi két assembly rutin segítségével a karakteres képernyőtartalmak MONITORból, BASIC-programból kimenthetők nyomtatóra.

Az első program nagybetűgrafika üzemmódban teszi lehetővé a képernyőtartalom kinyomtatását, a második C—16 MONITOR üzemmódban a D és M parancsok eredményeit nyomtatja ki.

Mindkét rutinhoz mellékelem a BASIC betöltőprogramot is.

1. Karakteres képernyő kinyomtatása

Az *1. programot* a kazetta és az RS—232 puffer helyére tettem, de az ugrási címek átírásával a BASIC-memória végére is áthelyezhető (MEMSIZ állítás!). A rutin lehetővé teszi üres soroknak a nyomtatási képből való kihagyását is (tömörít). Ha szükséges az üres sorok nyomtatása, akkor a rutin POKE118,1:SYS829-cel kell indítani. Tömöritett nyomtatáshoz POKE118,2-:SYS829-cel hívható BASIC-programból. A rutin inverz karaktereket is nyomtat.

1. program

```
0 REM KEPERNYO COPY ● NAGY TAMAS
1 FORI= 829TO 1070
2 READ J$:POKEI,DEC(J$)
3 NEXTI
60000  DATA A9,0C,85,72,A9,00,85,71
60001  DATA AA,A8,B1,71,20,5A,03,20
60002  DATA 69,03,C9,FF,D0,01,60,20
60003  DATA 78,03,4C,47,03,9D,2F,04
60004  DATA E0,27,F0,02,E8,60,20,86
60005  DATA 03,A6,71,60,A5,72,C9,0F
60006  DATA F0,01,60,C0,FF,F0,01,60
60007  DATA A9,FF,60,C8,C0,00,F0,01
60008  DATA 60,A5,72,18,69,01,85,72
60009  DATA 60,A2,00,BD,2F,04,C9,20
60010  DATA D0,08,E0,27,F0,12,E8,4C
60011  DATA 88,03,84,75,20,BA,03,20
60012  DATA DA,03,20,D1,03,A4,75,60
60013  DATA A5,76,C9,01,F0,01,60,84
60014  DATA 75,20,BA,03,20,F6,03,20
60015  DATA D1,03,A4,75,60,A9,04,A2
60016  DATA 04,A0,00,20,BA,FF,A9,00
60017  DATA 20,BD,FF,20,C0,FF,A2,04
60018  DATA 20,C9,FF,60,20,CC,FF,A9
60019  DATA 04,20,C3,FF,60,A2,00,BD
60020  DATA 2F,04,38,C9,80,B0,34,20
60021  DATA FC,03,20,D2,FF,E0,27,F0
60022  DATA 04,E8,4C,DC,03,20,F6,03
60023  DATA 60,A9,0D,20,D2,FF,60,38
60024  DATA C9,20,B0,04,18,69,40,60
60025  DATA 38,C9,40,B0,01,60,38,C9
60026  DATA 60,B0,04,18,69,20,60,18
60027  DATA 69,40,60,85,73,A9,12,20
60028  DATA D2,FF,A5,73,38,E9,80,20
60029  DATA FC,03,20,D2,FF,A9,92,4C
60030  DATA E7,03

OSSZEG  = 26907
KEZDOCIM= 829
VEGCIM  = 1070
```

A BASIC-betöltő

```
. 033D  A9 0C     LDA #$0C
. 033F  85 72     STA $72
. 0341  A9 00     LDA #$00
. 0343  85 71     STA $71
. 0345  AA        TAX
. 0346  A8        TAY
. 0347  B1 71     LDA ($71),Y
. 0349  20 5A 03  JSR $035A
. 034C  20 69 03  JSR $0369
. 034F  C9 FF     CMP #$FF
. 0351  D0 01     BNE $0354
. 0353  60        RTS
. 0354  20 78 03  JSR $0378
. 0357  4C 47 03  JMP $0347
. 035A  9D 2F 04  STA $042F,X
. 035D  E0 27     CPX #$27
. 035F  F0 02     BEQ $0363
. 0361  E8        INX
. 0362  60        RTS
. 0363  20 86 03  JSR $0386
. 0366  A6 71     LDX $71
. 0368  60        RTS
. 0369  A5 72     LDA $72
. 036B  C9 0F     CMP #$0F
. 036D  F0 01     BEQ $0370
. 036F  60        RTS
. 0370  C0 FF     CPY #$FF
. 0372  F0 01     BEQ $0375
. 0374  60        RTS
. 0375  A9 FF     LDA #$FF
. 0377  60        RTS
. 0378  C8        INY
. 0379  C0 00     CPY #$00
. 037B  F0 01     BEQ $037E
. 037D  60        RTS
. 037E  A5 72     LDA $72
. 0380  18        CLC
. 0381  69 01     ADC #$01
. 0383  85 72     STA $72
. 0385  60        RTS
. 0386  A2 00     LDX #$00
. 0388  BD 2F 04  LDA $042F,X
. 038B  C9 20     CMP #$20
. 038D  D0 08     BNE $0397
. 038F  E0 27     CPX #$27
. 0391  F0 12     BEQ $03A5
. 0393  E8        INX
. 0394  4C 88 03  JMP $0388
. 0397  84 75     STY $75
. 0399  20 BA 03  JSR $03BA
. 039C  20 DA 03  JSR $03DA
. 039F  20 D1 03  JSR $03D1
. 03A2  A4 75     LDY $75
. 03A4  60        RTS
. 03A5  A5 76     LDA $76
. 03A7  C9 01     CMP #$01
. 03A9  F0 01     BEQ $03AC
. 03AB  60        RTS
. 03AC  84 75     STY $75
. 03AE  20 BA 03  JSR $03BA
. 03B1  20 F6 03  JSR $03F6
. 03B4  20 D1 03  JSR $03D1
. 03B7  A4 75     LDY $75
. 03B9  60        RTS
. 03BA  A9 04     LDA #$04
. 03BC  A2 04     LDX #$04
. 03BE  A0 00     LDY #$00
. 03C0  20 BA FF  JSR $FFBA
. 03C3  A9 00     LDA #$00
. 03C5  20 BD FF  JSR $FFBD
. 03C8  20 C0 FF  JSR $FFC0
. 03CB  A2 04     LDX #$04
. 03CD  20 C9 FF  JSR $FFC9
. 03D0  60        RTS
```

```
. 03D1  20 CC FF  JSR $FFCC
. 03D4  A9 04     LDA #$04
. 03D6  20 C3 FF  JSR $FFC3
. 03D9  60        RTS
. 03DA  A2 00     LDX #$00
. 03DC  BD 2F 04  LDA $042F,X
. 03DF  38        SEC
. 03E0  C9 80     CMP #$80
. 03E2  B0 34     BCS $0418
. 03E4  20 FC 03  JSR $03FC
. 03E7  20 D2 FF  JSR $FFD2
. 03EA  E0 27     CPX #$27
. 03EC  F0 04     BEQ $03F2
. 03EE  E8        INX
. 03EF  4C DC 03  JMP $03DC
. 03F2  20 F6 03  JSR $03F6
. 03F5  60        RTS
. 03F6  A9 0D     LDA #$0D
. 03F8  20 D2 FF  JSR $FFD2
. 03FB  60        RTS
. 03FC  38        SEC
. 03FD  C9 20     CMP #$20
. 03FF  B0 04     BCS $0405
. 0401  18        CLC
. 0402  69 40     ADC #$40
. 0404  60        RTS
. 0405  38        SEC
. 0406  C9 40     CMP #$40
. 0408  B0 01     BCS $040B
. 040A  60        RTS
. 040B  38        SEC
. 040C  C9 60     CMP #$60
. 040E  B0 04     BCS $0414
. 0410  18        CLC
. 0411  69 20     ADC #$20
. 0413  60        RTS
. 0414  18        CLC
. 0415  69 40     ADC #$40
. 0417  60        RTS
. 0418  85 73     STA $73
. 041A  A9 12     LDA #$12
. 041C  20 D2 FF  JSR $FFD2
. 041F  A5 73     LDA $73
. 0421  38        SEC
. 0422  E9 80     SBC #$80
. 0424  20 FC 03  JSR $03FC
. 0427  20 D2 FF  JSR $FFD2
. 042A  A9 92     LDA #$92
. 042C  4C E7 03  JMP $03E7
```

2. program

```
0 REM MONITOR NYOMTATO ● NAGY TAMAS
1 FORI= 818TO 889
2 READ J$:POKEI,DEC(J$)
3 NEXTI
60000  DATA A9,03,A2,00,A0,00,20,BA
60001  DATA FF,A9,00,20,BD,FF,20,C0
60002  DATA FF,A9,0D,20,D2,FF,20,CC
60003  DATA FF,A9,03,20,C3,FF,60,20
60004  DATA 32,03,A9,04,A2,04,A0,00
60005  DATA 20,BA,FF,A9,00,20,BD,FF
60006  DATA 20,C0,FF,A2,04,20,C9,FF
60007  DATA 20,D2,FF,00,20,32,03,20
60008  DATA CC,FF,A9,04,20,C3,FF,00

OSSZEG  = 8574
KEZDOCIM= 818
VEGCIM  = 889
```

A program legfontosabb elemei:
Ø35A : Képernyősor tárolása
Ø386 : Vizsgálat : minden elem szóköz? Nem Ø397, igen Ø3A5
Ø397 : Nyomtatás (Ø3BA,Ø3DA,Ø3D1)
Ø3BA : Minden out a négyes nyomtatóra

Ø3DA : Beolvasás, vizsgálat (inverz), kódátalakítás, nyomtatás
Ø3D1 : Nyomtató lezárása
Ø3A5 : Vizsgálat : tömörítés?
Nem Ø3AC, igen Ø3AB (RTS)
Ø3AC : Soremelés (Ø3BA,Ø3F6,Ø3D1)
Ø3F6 : Soremelés
Ø369 : Vizsgálat : utolsó karakter?
Ø378 : Indexnövelés

2. MONITOR nyomtató

A C—16-os számítógép egyik fontos szolgáltatása a beépített MONITOR. Problémát jelent azonban, hogy ez a szoftver nem támogatja a nyomtatást. Ezért készítettem ezt a gépi rutint *(2. program)*. Hívása G Ø351-gyel történik. Ezt követően minden D és M parancs — a szokásos paraméterezéssel — eredménye a 4-es számú nyomtatóra megy. A szükséges adatok, programok kinyomtatása után ki kell kapcsolni a nyomtatórutint. Erre szolgál a rutin harmadik eleme, melynek hívása G Ø36E-vel történik. A program legfontosabb elemei:

Ø351 : Soremelő rutin (Ø332)
Ø332 : Képernyő megnyitása: SETFLS, SETNAM, OPEN soremelés, BSOUT, CLRCH, CLOSE
Ø354 : Nyomtató megnyitása SETFLS, SETNAM, OPEN, CKOUT, BSOUT
Ø36E : Soremelő rutin
Ø371 : Nyomtató lezárása : CLRCH, CLOSE

NAGY TAMÁS

A BASIC-betöltő

```
. 0332  A9 03     LDA #$03
. 0334  A2 00     LDX #$00
. 0336  A0 00     LDY #$00
. 0338  20 BA FF  JSR $FFBA
. 033B  A9 00     LDA #$00
. 033D  20 BD FF  JSR $FFBD
. 0340  20 C0 FF  JSR $FFC0
. 0343  A9 0D     LDA #$0D
. 0345  20 D2 FF  JSR $FFD2
. 0348  20 CC FF  JSR $FFCC
. 034B  A9 03     LDA #$03
. 034D  20 C3 FF  JSR $FFC3
. 0350  60        RTS
. 0351  20 32 03  JSR $0332
. 0354  A9 04     LDA #$04
. 0356  A2 04     LDX #$04
. 0358  A0 00     LDY #$00
. 035A  20 BA FF  JSR $FFBA
. 035D  A9 00     LDA #$00
. 035F  20 BD FF  JSR $FFBD
. 0362  20 C0 FF  JSR $FFC0
. 0365  A2 04     LDX #$04
. 0367  20 C9 FF  JSR $FFC9
. 036A  20 D2 FF  JSR $FFD2
. 036D  00        BRK
. 036E  20 32 03  JSR $0332
. 0371  20 CC FF  JSR $FFCC
. 0374  A9 04     LDA #$04
. 0376  20 C3 FF  JSR $FFC3
. 0379  00        BRK
```

Oktatóprogram PRIMO

Tanulmányaink során találkozhatunk olyan ismeretanyaggal, melynek megjegyzése sok energiát és időt igényel. Belátva, hogy a modern korban is szükség van a biflázásra, nekiülünk a tanulásnak. Az alábbi program segítségével ez a nehéz feladat esetleg hatékonyabbá és szórakoztatóbbá tehető. A gyakorlás során a gép kérdéseket tesz fel, értékeli a válaszokat és kiírja a helyes megoldást.

Egy játékban csak egy téma gyakorolható. A kérdések jó megválasztásával a prog-

```
4000 CLEAR 8000
4005 DIM R$(200),RR$(200)
4100 X(7)=116:X(6)=131:X(5)=148:X(4)=
     166:X(3)=176:X(2)=199:X(1)=223
4102 Y(7)=496:Y(6)=440:Y(5)=391:Y(4)=
     349:Y(3)=330:Y(2)=292:Y(1)=262
4104 N9(1)=12 : N9$(1)="ORDÖ"
4105 N9(2)=9 : N9$(2)="UGYES"
4106 N9(3)=6 : N9$(3)="SZES"
5980 REM ADATROGZITES >>>>>>>>>>>>>>>>
5981 CLS : PRINT : PRINT"MI A FELADAT?"
     : PRINT : PRINT : HE=0
5985 PRINT "1 SZAVAK SZALAGROL"
5986 PRINT "2 SZAVAK SZALAGRA"
5987 PRINT "3 UJ SZAVAK MEGADASA"
5988 PRINT "4 JATEK"
5989 PRINT "5 VEGE"
5990 PRINT "6 ELLENORZES"
5991 E$=INKEY$ : IF E$="" THEN 5991
5992 R=VAL(E$) : CLS
5993 ON R GOTO 6000,6010,6040,6060,599
     ,6070
5995 GOTO 5980
5996 END
6000 REM SZALAGROL
6001 PRINT "KAPCSOLD LEJATSZASRA A
     MAGNETOFONT!"
6003 OPEN "SZAVAK"
6004 INPUT# DB : INPUT# K$
6005 FOR I=1 TO DB
6006 INPUT# R$(I) : INPUT# RR$(I)
6007 NEXT I
6008 CLOSE
6009 GOTO 5980
6010 REM SZALAGRA
6015 PRINT "KAPCSOLD FELVETELRE A
     MAGNETOFONT!" : PRINT
6022 FOR I=1 TO 2000 : NEXT I
6023 CREATE "SZAVAK"
6024 PRINT DB
6025 PRINT K$
6026 PRINT# DB
6027 PRINT# K$
6028 FOR I=1 TO DB
6029 PRINT R$(I)
6030 PRINT RR$(I)
6031 PRINT# R$(I)
6032 PRINT# RR$(I)
6033 NEXT I
6034 FOR I=1 TO 30 : NEXT I
6035 CLOSE
6036 GOTO 5980
6040 REM UJ SZAVAK
6041 PRINT "MELYIK KERDES VALASZAIT";
     " GYAKOROLJUK?"
6043 PRINT "(FL.: KIT?-MIT?, KIVEL?-"
     "HIVEL?)" : INPUT K$
6045 PRINT# 13,0,"AZ UJ SZAVAK
     FELVETELET"
6047 PRINT# 12,0,"* BEIRASAVAL LEHET
     BEFEJEZNI."
6049 INPUT "HI AZ UJ SZO";E$
6050 IF E$="*" THEN 5980
6051 DB=DB+1
6052 IF DB>200 THEN 5980
6054 R$(DB)=E$ : PRINT K$ : INPUT
     RR$(DB)
6056 GOTO 6049
6060 REM JATEK
6064 GOTO 8000
6070 REM ELLENORZES >>>>>>>>>>>>>>>>>>
6072 FOR I=1 TO DB
6076 PRINT R$(I),RR$(I),I : PRINT
     "(NJ BE EGY KARAKTERT!"
6078 E$=INKEY$ : IF E$="" THEN 6078
6080 NEXT I
6082 GOTO 5980
8000 REM JATEK >>>>>>>>>>>>>>>>>>>>>>>
8001 FOR IK=1 TO 20
8005 Y=RND(DB)
8180 PRINT K$ : PRINT : PRINT "      "
     R$(Y) : PRINT
8210 INPUT C$
8220 IF C$<>RR$(Y) THEN 8300
8230 REM HELYES
8240 PRINT "     HELYES!" : HE=HE+1
8242 GOTO 8825
8800 REM HIBAS
8810 PRINT "    HIBAS!"
8820 PRINT "    HELYESEN:    "R$(Y)
8825 NEXT IK
8826 GOSUB 9079
8827 IF HE>N9(3) THEN GOSUB 9300
8829 GOTO 5980
8830 END
9079 REM DALLAM -EGY KIS MALAC .....
9080 DATA 7,M,2,S,2,R,2,M,2,D,2,D,2,D,
9081 DATA 7,M,2,B,2,R,2,M,2,D,2,D,2,D,
9082 DATA 8,M,2,S,2,R,2,M,2,D,2,B,2,M,2
     ,M,2
9084 DATA 7,L,2,L,2,M,2,M,2,S,2,S,2,S,
9085 RESTORE
9086 FOR K=1 TO 4
9089 READ I9
9090 FOR I=1 TO I9
9091 READ A$,B
9092 IF A$="D" THEN A=1
9093 IF A$="R" THEN A=2
9094 IF A$="M" THEN A=3
9095 IF A$="F" THEN A=4
9096 IF A$="S" THEN A=5
9097 IF A$="L" THEN A=6
9098 IF A$="T" THEN A=7
9099 BEEP X(A),Y(A)/(B*1.25)
9100 NEXT I
9101 FOR I=1 TO 80 : NEXT I
9102 NEXT K
9103 RETURN
9300 REM A LEGJOBBAK >>>>>>>>>>>>>>>>
9308 INPUT "IRD BE A NEVED! (MAX. 10
     BETU.)";N9$(4)
9310 N9$(4)=LEFT$(N9$(4)+"          "
     ,10) : N9(4)=HE : CLS
9312 FOR I9=1 TO 4 : FOR J9=I9 TO 4
9313 IF N9(J9)<=N9(I9) THEN 9317
9314 C9=N9(J9):N9(J9)=N9(I9):
     N9$(J9)=N9$
9316 C9=N9(J9):N9(J9)=N9(I9):N9(I9)=C9
9317 NEXT J9 : NEXT I9
9318 PRINT "    A LEGJOBBAK" : PRINT :
     PRINT
9320 FOR I9=1 TO 3
9321 PRINT N9(I9),N9$(I9)
9322 NEXT I9
9323 FOR I9=1 TO 8000 : NEXT I9
9324 RETURN
```

ramot sokoldalúan lehet használni. Néhány példa a lehetséges kérdésekre: Kivel, mivel? Mit csinált? Hogy van angolul? Mikor született? Mi a fővárosa? Hány vegyértékű? Mikor épült? Ki építette? Miből készül?

A tanulás a téma kiválasztásával és a kérdés meghatározásával kezdődik, majd meghatározzuk az adathalmazt. Az adatok rögzítését és ellenőrzését a menüből irányíthatjuk.

Az egy menetben rögzíthető feladatok száma a 4000-es sorban levő CLEAR utasítás utáni számmal állítható be gépünk méretének megfelelően, de a program 200 feladatnál többet nem fogad el.

SOMOGYI GYÖRGY

Problémát okoz Önnek, megbízhatatlan a hálózati feszültség?

AZ ASM—250
SZÜNETMENTES ÁRAMFORRÁS

hálózat kimaradása esetén megszakítás nélkül min. 30 perc időtartamig biztosítja a 220 V, 50 Hz-es kimenő feszültséget. Névleges teljesítménye: 250 VA.

Bővebb felvilágosítással szolgál:

ERFI

Vállalkozási Iroda,
1027 Budapest,
Medve u. 25/29.
Telefon: 354-140, 359-740
Telex: 22 5982 erfi

Hangszerprogram HT géphez FORTH nyelven

Hazánkban minden középiskolában van HT—1080Z típusú számítógép, amely a TII által az iskoláknak nemrég megküldött FORTH-kazetta és programozói kézikönyv felhasználásával már ezen a — BASIC-nél tömörebb (és gyorsabban futó) programok készítésére alkalmas — nyelven is programozható.

Az itt ismertetett program a HT géphez adaptált FORTH, a zFORTH alkalmazásához kíván kedvet csinálni. A zFORTH betöltése után egyszerűen be kell írni a programot a gépbe, majd H NEW LINE paranccsal indítható a futása. Hatására a számítógép egyszerű hangszerré alakul: a felfele nyil billentyűt benyomva C, a Q billentyű hatására D hang szólal meg, majd a billentyűsoron végighaladva, a P billentyűvel bezárólag a skála valamennyi egész hangját képezhetjük. A félhangokat előállító billentyűk a legfelső sorban helyezkednek el, hasonlóan a zongora fekete billentyűinek elrendezéséhez. A hangszer az említett billentyűk lenyomását követően folyamatos, állandó intenzitású hangot hoz létre, amely bármely, a hangok képzésénél fel nem használt billentyű megnyomásával szüntethető meg.

Az indításnál használt H szó lényegében az egész programot tartalmazza. A definíciójában elöl szereplő ALAP szó SOUND utasításai nyomán a hanggenerátor 7. regiszterébe 254, a nyolcadikba 10 kerül. Az első szám a hanggenerátor A csatornájának kizárólagos engedélyezését, a második „környezetkímélő" hangerő beállítását végzi. Ha maximális hangerőt kívánunk elérni, akkor a 10 helyébe 15 írandó. Az utána következő BEGIN az UNTIL-lel ciklust hoz létre. Az e ciklus magjában található KEY a verembe helyezi az éppen megnyomott billentyűhöz tartozó ASCII kódot. Ezt követően a B szóban definiált CASE struktúra az előbb elhelyezett szám alapján kiválasztja a hanggenerátor Ø. regiszterébe töltendő számot. Ha a hangképzésben nem használt billentyűt nyomunk le, akkor ez a szám — a CASEND előtt elhelyezett Ø révén — Ø lesz. A SOUND utáni Ø biztosítja, hogy a BEGIN—UNTIL ciklus a gép kikapcsolásáig megszakítás nélkül ismétlődjék.

Az ALAP, B,H általunk létrehozott szavak, így ízlés szerint másokkal is helyettesíthetők. A többiek közvetlenül nem, mert a zFORTH alapszókészletéhez tartoznak.

A program az igényektől függően sokoldalúan bővíthető (hangterjedelem bővítése, ütőhangzás stb.). Ez esetben is megfigyelhető a FORTH előnye: hangszerünk lényegesen gyorsabb hangátmenetekre képes a BASIC-programozással előállíthatónál.

PÁL LÁSZLÓ

A program

```
(HANGSZER C:FELFELE NYIL —
FISZ:PROGRAM)
(EGYEB BILLENTYUVEL ELHALL-
GAT)
: ALAP 254 7 1Ø 8 SOUND SOUND ;
: B CASE 91 OF 2Ø8 ENDOF 81 OF
  186 ENDOF 87 OF 165 ENDOF 69
  OF 156 ENDOF 82 OF 139 ENDOF
  84 OF 124 ENDOF 89 OF 11Ø ENDOF
  85 OF 1Ø4 ENDOF 73 OF 93 ENDOF
  79 OF 83 ENDOF 8Ø OF 78 ENDOF
  49 OF 197 ENDOF 5Ø OF 175 ENDOF
  52 OF 148 ENDOF 53 OF 131 ENDOF
  54 OF 116 ENDOF 56 OF 98 ENDOF
  57 OF 88 ENDOF 58 OF 74 ENDOF
  Ø CASEND ;
: H ALAP BEGIN KEY B Ø SOUND Ø
  UNTIL ;
```

A szerkesztő megjegyzése. A lapunk 1986/9. számában megjelent „Hangkeltés HT gépen FORTH-ban" c. cikk programja nem zFORTH-ban íródott. Itt azért nincs szükség gépi nyelvű primitívek írására, mert azok, amelyek a HT speciális szolgáltatásainak alkalmazásához szükségesek, már be vannak építve.

Tervezőintézet országos számítógépes grafikai nyilvántartási rendszerek fejlesztéséhez keres rendszerszervező, tervező és programozó szakembereket.
Jelentkezés: 569-122/218-as mellék

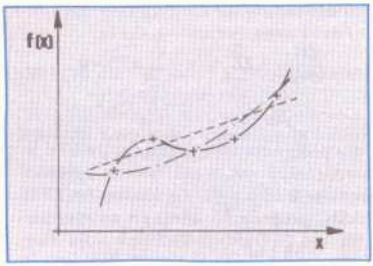
Pontsor közelítése polinommal

Függvény típusú mérési eredményeket ábrázoló pontsor közelítésére többek között polinomokkal is próbálkozhatunk. Ennek legegyszerűbb módja a pontsorra illeszkedő ún. Lagrange-polinom. Ez azonban mérési hibákkal terhelt pontsor közelítésére nemigen alkalmas, hiszen a véletlen hibák hatását éppen úgy lehet „kisimítani", ha a görbe nem megy át mindegyik ponton *(1. ábra).*

Vezessük be az alábbi jelöléseket:
mérési adatpárok: x_i, y_i $(i = 1, 2, \ldots, m)$
közelítő polinom: $p(x) = \sum_{j=0}^{n} a_j x^j$ $(n \le m-1)$

Az n = m-1 eset a Lagrange-polinom, például két pontra fektetett egyenes, három pontra fektetett parabola stb.

Mérési adatok polinomos közelítésekor két eset lehetséges:



1. ábra: Mérési hibákkal terhelt pontsor polinomos közelítése

+ mért pontok

--- regressziós egyenes

——— illeszkedő polinom

—.— valódi függvény

2. ábra

1. Ismerjük az összefüggés n fokszámát és keressük az a_j együtthatókat.

2. A fokszámot sem ismerjük.

Az első esetben a Lagrange-polinomot illesztő program egyszerre csak m = n + 1 mérési pont használatát engedi meg. A második esetben pedig végképp használhatatlan, hiszen ha ötletszerűen kiragadott pontokra illesztünk Lagrange-polinomot és a kiragadott pontok számán keresztül a polinom fokszámát változtatjuk, akkor semmilyen módon nem tudjuk eldönteni, hogy melyik a legjobb közelítés. Tehát mindenképpen változtatható fokszámú polinommal közelítő programra van szükségünk.

A közelítés gondolatmenete a következő. Az a_j együtthatókat a H négyzetes hiba minimumának megkeresésével határozzuk meg.

```
 1 CLS
 2 PRINT AT 5,12;"POLINOM"
 3 PRINT AT 6,11;""
 4 PRINT AT 12,1;"MIT KERSZ?"
 5 PRINT AT 15,4;"A PROGRAM MAGYARAZATAT   (M/"
 6 PRINT AT 17,4;"A PROGRAM FUTTATASAT     (F/"
 7 INPUT A$
 8 IF A$="F" THEN GOTO 15
 9 IF A$="M" THEN GOTO 350
11 PRINT " NA MOST MELYIKET?"
12 GOTO 7
13 CLS
14 PRINT "MERESI PONTOK SZAMA= ";
15 INPUT N
16 PRINT N;,,,
17 PRINT "MERESI PONTOK ES ADATOK:",,,
18 DIM X(N)
19 DIM Y(N)
20 FOR J=1 TO N
21 PRINT J;".PONT= ";
22 INPUT X(J)
23 PRINT X(J)
24 PRINT J;".ADAT= ";
25 INPUT Y(J)
26 PRINT Y(J)
27 NEXT J
28 PAUSE 100
29 CLS
30 PRINT "A POLINOM FOKSZAMA= ";
32 INPUT P
33 PRINT P;
34 IF P<N-1 THEN GOTO 55
35 IF P=N-1 THEN GOTO 58
36 PRINT ,,"TUL NAGY FOKSZAM ENNYI PONTHOZ."
37 PRINT ,,"ADJ AU FOKSZAMOT.",,,
39 GOTO 30
58 PRINT ,,"MINDEGYIK PONTON AT FOG MENNI."
59 PAUSE 100
60 PAGE
65 DIM H(P+1,P+1)
70 FOR S=0 TO P
75 FOR I=0 TO N
76 FOR J=1 TO N
77 IF X(J)=0 THEN GOTO 85
80 IF S+1=2*INT (S+1)/2) THEN GOTO 85
81 LET H(S+1,I+1)=H(S+1,I+1)-(ABS X(J))**(S+1)
82 GOTO 86
85 LET H(S+1,I+1)=H(S+1,I+1)+(ABS X(J))**(S+1)
86 NEXT J
87 LET H(I+1,S+1)=H(S+1,I+1)
88 NEXT I
89 NEXT S
115 DIM B(P+1)
120 FOR S=0 TO P
125 FOR J=1 TO N
126 IF X(J)=0 THEN GOTO 134
129 IF S=2*INT (S/2) THEN GOTO 134
130 LET B(S+1)=B(S+1)-Y(J)*(ABS X(J))**S
132 GOTO 135
134 LET B(S+1)=B(S+1)+Y(J)*(ABS X(J))**S
135 NEXT J
136 NEXT S
137 DIM N(P+1,P+1)
138 DIM C(P+1)
139 FOR S=1 TO P+1
140 FOR I=1 TO P+1
142 LET N(S,I)=H(S,I)
144 NEXT I
145 LET C(S)=B(S)
146 NEXT S
151 PRINT ,,"A POLINOM EGYUTTHATOI:",
155 DIM A(P+1)
170 DIM Z(P+1)
180 FOR I=1 TO P+1
185 FOR S=1 TO P+1
186 IF S=I THEN GOTO 210
188 LET Z(S)=H(S,T)/H(I,T)
190 FOR I=1 TO P+1
195 LET H(S,I)=H(S,I)-H(I,I)*Z(S)
200 NEXT I
205 LET B(S)=B(S)-B(T)*Z(S)
210 NEXT S
215 NEXT T
220 FOR S=P+1 TO 1 STEP -1
225 LET A(S)=B(S)/H(S,S)
230 PRINT ,,"A(";S-1;")= ";A(S)
235 NEXT S
238 DIM W(N)
239 LET M=0
240 FOR J=1 TO N
241 LET W(J)=0
242 FOR S=1 TO P+1
244 IF X(J)>=0 THEN GOTO 252
246 IF S-1=2*INT ((S-1)/2) THEN GOTO 252
248 LET W(J)=W(J)-A(S)*(ABS X(J))**(S-1)
250 GOTO 254
252 LET W(J)=W(J)+A(S)*(ABS X(J))**(S-1)
254 NEXT S
256 LET M=M+(ABS (Y(J)-W(J)))**2
258 NEXT J
260 PRINT ,,"NEGYZETES HIBA = ";M
270 SLOW
285 PRINT ,,,"KERSZ MAGOLATOT?      (I/N)"
286 INPUT B$
287 IF B$="N" THEN GOTO 312
288 IF B$="I" THEN GOTO 300
289 PRINT ,,"IGEN VAGY NEM?"
290 GOTO 286
300 CLS
312 PRINT ,,,"UJ FOKSZAM?           (I/N)"
314 INPUT C$
316 IF C$="N" THEN GOTO 346
318 IF C$="I" THEN GOTO 29
320 PRINT ,,"IGEN VAGY NEM?"
322 GOTO 314
346 PRINT ,,,"VEVETKEZ0?            (I/"
347 INPUT C$
348 IF C$="I" THEN GOTO 400
349 GOTO 13
350 CLS
352 PRINT "EZ A PROGRAM MERESI HIBAKKAL    TERHELT PONTSORT KOZELIT VA-    LASZ
THATO FOKSZAMU POLINOMMAL.   A POLINOM FOKSZAMA KISEBB KELL  HOGY LEGYEN A MERESI
 PONTOK SZA-MANAL. HA CSAK EGGYEL KISEBB, A POLINOM ATMEGY MINDEGYIK PONTON.A ME
RESI PONTOK ES ADATOK UTAN  A PROGRAM A VALASZTOTT FOKSZA-  MOT KERI. HA EZ MEGF
ELELO, KI-  SZAMOLJA ES KIIRJA A POLINOM    EGYUTTHATOIT. A(1) AZ 1-EDIK    MATV
ANYON LEVO TAG SZORZOJA."
354 PRINT "AZ ADATPAROK UJBOLI MEGEPELESE  NELKUL TOBB FOKSZAM IS KIPRO-   BALH
ATO. LEUJJOBBNAN AZ KOZELIT,  AMELYIK A LEGKISEBB NEGYZETES  HIBAT ADJA."
360 PRINT ,,,"  KEZDHETJUK?          (I/"
370 INPUT B$
380 IF B$="I" THEN GOTO 400
390 GOTO 13
```

```
A POLINOM FOKSZAMA: 1

A POLINOM EGYUTTHATOI:

A(1)= 51.352941

A(0)= -6.8529412

NEGYZETES HIBA = 9436.7647

KERSZ MASOLATOT?     >I/N<

A POLINOM FOKSZAMA: 2

A POLINOM EGYUTTHATOI:

A(2)= 5

A(1)= 41.352941

A(0)= -44.352941

NEGYZETES HIBA = 3811.7647

KERSZ MASOLATOT?     >I/N<

A POLINOM FOKSZAMA: 3

MINDEGYIK PONTON AT FOG MENNI.

A POLINOM EGYUTTHATOI:

A(3)= 3

A(2)= -4

A(1)= 5.0000001

A(0)= -2.0000003

NEGYZETES HIBA = 9.0594199E-1

KERSZ MASOLATOT?     >I/N<

A POLINOM FOKSZAMA: 0

A POLINOM EGYUTTHATOI:

A(0)= 44.5

NEGYZETES HIBA = 99099

KERSZ MASOLATOT?     >I/N<
```

3. ábra

$$H=\sum_{i=1}^{m}(y_i-p(x_i))^2$$

$$H=\sum_{i=1}^{m}\Big(y_i-\sum_{j=0}^{n}a_jx_i^j\Big)^2$$

A négyzetes hiba ott a legkisebb, ahol az összes a_j szerinti deriváltja nulla. Az a_k szerinti derivált:

$$\frac{dH}{da_k}=-2\sum_{i=1}^{m}x_i^k\Big(y_i-\sum_{j=0}^{n}a_jx_i^j\Big)=0$$

$$\sum_{j=0}^{n}a_j\Big(\sum_{i=1}^{m}x_i^{j+k}\Big)=\sum_{i=1}^{m}x_i^ky_i$$

Ez pedig nem más, mint az a_j-kre mint ismeretlenekre vonatkozó lineáris egyenletrendszer k-adik sora. Az általam írt ZX81 BASIC program *(2. ábra)* ezt az egyenletrendszert a Gauss-módszerrel oldja meg. A program 1 ... 12 és 350 ... 390 sorai elhagyhatók, ha a magyarázó szövegre nincs szükség.

Ha a keresett összefüggés fokszámát nem ismerjük, akkor az adatpárok újbóli begépelése nélkül többfajta fokszámú polinomot is kipróbálhatunk. A legjobban az közelíti meg a keresett összefüggést, amelyik a legkisebb négyzetes hibát adja.

Szemléltetés céljára az

$$f(x)=3x^3-4x^2+5x-2$$

függvényt kielégítő néhány adatpárt adtam meg a programnak. A futtatások eredményei a *3. ábrán* láthatók. A nulladfokú polinom a függvényértékek számtani átlaga, az első fokú a regressziós egyenes stb. a program láthatóan igen pontosan meghatározta az eredeti összefüggést.

A téma iránt érdeklődőknek az alábbi szakirodalmat ajánlom:

[1] Boros Andor: Mérésértékelés. Bp. 1982. Műszaki Könyvkiadó.

[2] Tolnai Gábor István: Görbére illeszkedő polinom kiszámítása. Mikroszámítógép Magazin, 1985/5. szám.

[3] Dr. Ambrózy András — Jávor András: Mérésadatok kiértékelése. Bp. 1976. Műszaki Könyvkiadó.

PATAKI BÉLA

COMMODORE 64

Függvények bemenetben való megadásáról többször volt már szó a Mikroszámítógép Magazin hasábjain. Sinclair Spectrum gépen a probléma igen egyszerűen megoldható a VAL BASIC függvény segítségével. Egyéb típusú gépeken BASIC-programokkal igyekeztek megoldani a kérdést.

Egy ilyen program megírásának alapgondolata az, hogy létrehozzunk egy, a beolva-

1. lista

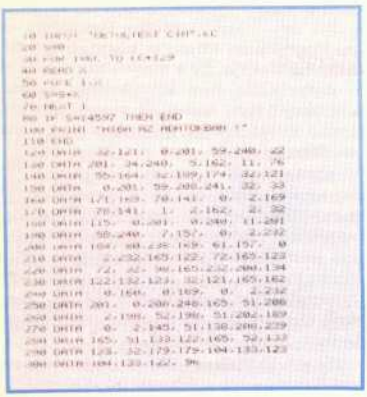
```
2
0334                       .OPT OO,P5
0334           BYTE0    =  "F"
0334           BYTE1    =  "N"
0334           BYTE2    =  "="
0334           LINE     =  #A562
0334           TOKEN    =  #A579
0334           DEFFN    =  #B3B2
0334           CHRGOT   =  #0079
0334           CHRGET   =  #0073
0334           STROUT   =  #AB21
0334           ERROR    =  #A437
0334           DIASTR   =  #AEBD
0334                    *= #0334
0334 20 79 00           JSR CHRGOT
0337 C9 3B              CMP ##3B
0339 F0 16              BEQ PUFFER
033B C9 22              CMP ##22
033D F0 05              BEQ STRING
033F A2 0B     SYNTAX   LDX ##0B
0341 4C 37 A4           JMP ERROR
0344 20 BD AE  STRING   JSR DIASTR
0347 20 79 00           JSR CHRGOT
034A C9 3B              CMP ##3B
034C D0 F1              BNE SYNTAX
034E 20 21 AB           JSR STROUT
0351 A9 46     PUFFER   LDA #BYTE0
0353 8D 00 02           STA #0200
0356 A9 4E              LDA #BYTE1
0358 8D 01 02           STA #0201
035B A2 02              LDX ##02
035D 20 73 00  READN    JSR CHRGET
0360 C9 00              CMP ##00
0362 F0 0B              BEQ CONT
0364 C9 3A              CMP ##3A
0366 F0 07              BEQ CONT
0368 9D 00 02           STA #0200,X
036B E8                 INX
036C B8                 CLV
036D 50 EE              BVC READN
036F A9 3D     CONT     LDA #BYTE2
0371 9D 00 02           STA #0200,X
0374 E8                 INX
0375 A5 7A              LDA #7A
0377 48                 PHA
0378 A5 7B              LDA #7B
037A 48                 PHA
037B 20 62 A5           JSR LINE
037E E8                 INX
037F C8                 INY
0380 86 7A              STX #7A
0382 84 7B              STY #7B
0384 20 79 A5           JSR TOKEN
0387 A2 00     MOVE     LDX ##00
0389 A0 00              LDY ##00
038B BD 00 02  CONT2    LDA #0200,X
038E E8                 INX
038F C9 00              CMP ##00
0391 D0 F8              BNE CONT2
0393 A5 33     CONT3    LDA #33
0395 D0 02              BNE FORWARD
0397 C6 34              DEC #34
0399 C6 33     FORWARD  DEC #33
039B CA                 DEX
039C BD 00 02           LDA #0200,X
039F 91 33              STA <#33>,Y
03A1 8A                 TXA
03A2 D0 EF              BNE CONT3
03A4 A5 33              LDA #33
03A6 85 7A              STA #7A
03A8 A5 34              LDA #34
03AA 85 7B              STA #7B
03AC 20 B3 B3           JSR DEFFN
03AF 68                 PLA
03B0 85 7B              STA #7B
03B2 68                 PLA
03B3 85 7A              STA #7A
03B5 60                 RTS
```


Személyiszámítógép-javítás, karbantartás közületeknek, magánszemélyeknek.

Egyedi megrendelés alapján kiegészítő berendezések gyártása.
Pl.: Sinclair fényceruza, Joystic Interface,
oktatási intézmények kabinet kialakítása.

Pásztor Ferenc személyiszámítógép-javító és -karbantartó kisiparos.
Szolnok, Mátyás király u. 2. V/4.

Függvénybemenet



2. lista

3. ábra

sandó függvénynek megfelelő karakterláncot. Ezután hívjuk a gép BASIC-fordítóprogramjába beépített DEFFN rutint.

A programot *(1. lista)* úgy írtam meg, hogy az minden változtatás nélkül betölthető és futtatható bármilyen szabad RAM-területen, és felhasználja a ROM-ban megtalálható rutinokat.

Az újonnan beépített utasítás a következőképpen hívható:

SYS DE ; F(X) vagy

SYS DE "F(X)="; F(X)

ahol DE a betöltési cím. Látható, hogy az utasítás megengedi egy karakterlánc megadását, ami híváskor megjelenik a képernyőn.

A program részletes bemutatása előtt néhány szó a DEFFN rutin működéséről. Ha a BASIC-fordítóprogram működés közben a DEF kulcsszó tokenjét találja, „felírja" a RAM-ba a függvény nevét és annak a memóriaterületnek a kezdőcímét, ahol a függvény szövege található.

Billentyűzetről beolvasott függvény esetén a függvény szövegét a bemeneti pufferből előbb egy nem felülíródó RAM-területre kell helyezni, ami jelen esetben a sztringterület. Ennek a területnek az érdekessége, hogy a szabad RAM-terület végén található, lefelé bővül, és ide kerülnek a bemenetben megadott szöveges változók.

A program működése:

- 20—120 A felhasznált ROM-rutinok címei.
- 140—160 Első karakter beolvasása; ha ";" akkor a függvény neve következik.
- 170—180 Ha """ akkor a kiírandó karakterlánc következik.
- 190—200 Ha egyik sem, akkor hibaüzenet.
- 210—250 Karakterek leolvasása és kiírása a képernyőre.
- 260—300 FN elhelyezése a bemeneti pufferbe.
- 310—390 A függvény nevének leolvasása és elhelyezése a bemeneti pufferbe.
- 400—460 A BASIC-szövegmutató (S7A,$7B) mentése a verembe.
- 470—510 A függvény beolvasása a billentyűzetről.
- 520 A bemeneti pufferben levő szöveg tokenizálása.
- 530—710 A függvény tokenizált szövegének áthelyezése a sztringterületre.
- 720 A DEFFN rutin hívása.
- 730—770 A szövegmutató visszaállítása.

A programot a *2. lista* programjával tölthetjük be a kiválasztott RAM-területre. A *3. lista* új utasítás felhasználására ad példát.

BOÉR ÁDÁM—NAGY LÁSZLÓ

Játsszunk számítógépet!

Alekszandr Dnyeprov szovjet sci-fi író Játék című novellájában sajátos módon próbálják eldönteni azt a kérdést, hogy gondolkoznak-e a számítógépek. Egy matematikai konferencia mintegy félezer résztvevőjét szigorú rendszer szerint felállítják egy sportpálya gyepén, mindenki bizonyos szabályok szerint fogad, feldolgoz és továbbít számokat. Mint utóbb kiderül, portugálról fordítottak oroszra egy mondatot, szimulálva egy fordítóprogram munkáját.

Az ötlet rendkívül szellemes, érdemes foglalkozni vele egy kicsit. Nem az eredeti kérdésfeltevés értelmében, hiszen nem hinném, hogy bárki is a jelenlegi gépekről tételezne fel értelmes gondolkodást. A filozófiai és szaktudományi vita inkább ennek elvi (gyakorlati?) lehetségességéről folyik. Az AI- (Artifical Intelligence = mesterséges intelligencia) kutatások jelenlegi eredményei — társalgó, logikai játékokat megoldó, fordító stb. programok — inkább arra mutatnak rá, hogy mi mindent *ne* tekintsünk feltétlenül a gondolkodás jelének.

Dnyeprov ideáját, az emberekből felépített számítógépet viszont rendkívül jól fel lehet használni az oktatásban. Jómagam a NIM-játék algoritmusát "hangszereltem" diákokra. Több ízben tartottam sikeres bemutatót az élő számítógéppel, majd megbeszéltük a tapasztaltakat. Roppant érdekes volt a gyerekek reagálása arra, hogy a személlők, sőt a végrehajtók számára is áttekinthetetlen számítási műveletek eredményeképpen a "gép" milyen magabiztosan nyeri meg a játékot.

A NIM szabályai a következők. Gyufaszálak, pénzérmék stb. csoportjaiból kell két játékosnak felváltva valamennyit elvenni. Egyszerre csak egy csoportból lehet húzni, de akár az egész csoportot is elvehetjük. Az győz, akinek az utolsó elem jut. Szokás úgy játszani, hogy az veszítsen, akié az utolsó elem. Ez elvileg nem nagyon befolyásolja az algoritmust, viszont számunkra fölösleges bonyodalmakkal járna.

Az említett demonstrációs célokhoz éppen egy ilyen játék tűnik a legmegfelelőbbnek. Elég bonyolult ahhoz, hogy ne lehessen túl könnyen kitalálni a helyes lépéseket, de eléggé egyszerűnek látszik, tehát felkelti az érdeklődést. Másrészt algoritmusa könnyen programozható, akár élő "alkatrészekre" is.

A bemutatóhoz 5-7 vállalkozó elegendő. Két személy — "A" és "B" végzi a munka nehezét, őket a jobban számolók közül kell kiválasztani. A többiek a tárolót alkotják, számuk határozza meg, hogy mekkora csoportokat tud kezelni a "gép".

"A" feladatai

Megszámolja az egyes csoportokban levő elemek számát. Ha már mindegyik üres, azt mondja: "vesztettem". Egyébként a csoportok elemszámát rendre közli "B"-vel. Az utolsó szám után azt mondja "B"-nek, hogy "KÉSZ", majd kap "B"-től egy számot. Ha ez nulla, akkor a legnagyobb csoportból elvesz egyet. Ha nem, akkor az ennyi elemet tartalmazó csoportból kell venni. Sorra kér egy-egy számot "5", "4", "3", "2", "1"-től. Az "5"-től kapott számot megszorozza kettővel, majd hozzáadja a "4"-től kapott számot. Az eredményt szorozza kettővel, s hozzáadja a következőtől kapott számhoz stb. A "B"-től kapott számú elemet tartalmazó csoportban az előző műveletek eredményéül kijött számú elemet kell hagyni!

"B" feladatai

"A"-tól vár számokat, amelyeket egyenként a következőképpen dolgoz fel. Elosztja kettővel maradékosan, s odamegy az "1"-eshez. Ha a maradék egy, akkor megmondja "1"-nek az "A"-tól kapott eredeti számot. Most az osztás eredményét osztja kettővel, s a maradéktól függ, hogy megmondja-e "2"-nek az "A"-tól kapott számot stb. egészen "5"-ig.

Amikor "A" azt mondja, hogy "KÉSZ", akkor odamegy az "5"-öshöz és kér tőle egy számot. Ha ez nulla, akkor a "4"-estől kér stb., amíg csak nem nullát kap, vagy már az "1"-estől is nullát kapott. Ha kapott nem nulla számot, akkor ezt ugyanúgy, mintha az "A"-tól származna, feldolgozza. Végül megmondja "A"-nak ezt a számot.

"1", "2", "3", "4", "5" feladatai

Sorra kap "B"-től számokat. Számolnia kell, hogy hány számot kapott és meg kell jegyezni közülük a legnagyobbat. Ha "B" számot kér tőle, akkor nullát kell mondania, amennyiben páros számú számot kapott, és a kiválasztott legnagyobbat, ha páratlant. Ezek után "B" adhat neki még egy számot. Ezt is bele kell számítania a "B"-től kapott számok számába, s ha ez páros, akkor nullát, ha páratlan, akkor egyet kell adni "A"-nak, amennyiben az számot kér tőle.

LOVRICS LÁSZLÓ

Milyen a jó számítógépes játék?

Az UNESCO—CODIESEE szófiai szakértői értekezleten a tanulói kreativitás fejlesztése, ezen belül a számítógépes oktatási játékok kerültek napirendre.

A résztvevő országok (Bulgária, Jugoszlávia, Görögország, Magyarország, Spanyolország és Málta) nemcsak a számítógépes játékok oktatási felhasználásáról számoltak be, hanem összefoglalták a számítástechnika és a számítógéppel segített oktatás bevezetésének első hazai tapasztalatait is. Természetesen a számítógépes játékok iskolai alkalmazásának elemzéséhez elsősorban számítógépre, programokra és felkészült tanárokra van szükség. Igazán büszkén számolhattunk be a magyar iskola-számítógépes program eddigi eredményeiről, a gépek magyarországi oktatási felhasználásáról. Örömmel láttuk, hogy Marx György *A természet játékai* című könyvének angol fordítása és a benne levő programok már az UNESCO képviselője számára sem jelentettek újdonságot, mivel ajánlotta azokat. Számos, angol nyelven megírt magyar játékprogramot vihettünk magunkkal, melyeket be is mutattunk.

A jó számítógépes játék. Ahhoz, hogy a számítógépet alkalmazni kell az oktatásban, hogy a jövőben az általános műveltség elképzelhetetlen lesz informatikai alapműveltség nélkül, hogy mindennapi életünk szerves részét képezi majd a számítógép — nem fér kétség. De vajon szükség van-e oktatásban számítógépes játékokra? Mennyire lehet eredményesen tanulni a számítógépes játékokkal? Melyik életkorban, melyik korcsoport számára hasznos igazán a játék? Ilyen és ehhez hasonló kérdéseket elemeztek, vitattak a tanácskozás résztvevői.

A számítógépek sokféle iskolai felhasználása között egyértelműen helyet kapnak a számítógépes didaktikai játékok is, melyek a számítógépek iskolai alkalmazását még hasznosabbá teszik. A jó oktatási célú játék a készségek egész sorát fejleszti ki, új ismereteket közölhet anélkül, hogy elvontabb tudományos vagy programozási ismereteket feltételezne. A didaktikai játékok közül a legfontosabb szerep a *szimulációs* játékoknak jut, melyek kiválóan alkalmasak arra, hogy számítógépre vigyük őket. Jó döntésjátékokra és gyakorlójátékokra is szükség van; utóbbiakra különösen kisiskolás korban.

Az igazán jó számítógépes játék szellemes, ötletes, jól érthető szabályai vannak, könnyen kezelhető, és értékeli is a tanuló tudását vagy ügyességét. A pusztító, romboló célú, becsapós játékok és a hazárdjátékok alkalmazását szigorúan kerülnünk kell! A jó számítógépes játékkal a gyermek tanulási motivációja nő, logikai képessége fejlődik, megismeri stratégiák alkalmazását is, és a géppel való kommunikáció során felhasználói szintű programozási ismereteket is szerezhet.

A tanácskozás résztvevői elhatározták, hogy minden tagország összegyűjt 5—10 számítógépes didaktikai játékot, ezeket angol nyelvre lefordítva sokszorosítja, és elküldi a többi tagországnak. A programokhoz olyan dokumentációt is mellékel, amely bemutatja a játékot és annak pedagógiai célját. Mivel az iskolai számítógépek országonként különbözők, a kapott segédanyagok alapján minden ország átdolgozza majd a programokat saját iskola-számítógépére, és kipróbálja azokat az oktatási folyamatban. Az így nyert érdekes és értékes oktatási tapasztalatokat összegyűjtve ismét megküldik egymásnak.

Nagy tetszést aratott az a javaslat, hogy olyan találkozót vagy konferenciát kellene szervezni, amelyen nemcsak tanárok, hanem diákok is részt vehetnek; a feladat egyrészt a meglévő didaktikai játékok pedagógiai értékelése lenne, másrészt — és itt jutnának fontos szerephez a részt vevő tanulók — új, számítógépes oktatási játékok alkotása, az ötletgyűjtéstől kezdve egészen a konkrét programírásig. Elhangzott olyan javaslat is, hogy érdemes lenne megvizsgálni a kereskedelmi forgalomban kapható játékprogramokat, és elemezni oktatási felhasználhatóságukat. Szükségesnek látnák a résztvevők azt is, hogy minden iskola-számítógépekkel kapcsolatos anyagról (könyvek, tankönyvek, kísérletek, tanulmányok, kutatások, programok) vagy rendezvényről (konferencia, tanácskozás, anket) a jövőben kölcsönösen tájékoztassák egymást. Jó lenne az oktatási szoftvert író csoportok között nemzetközi koordinációt kialakítani.

KŐRÖSNÉ MIKIS MÁRTA

A számítógép, mint a valóságot közvetítő eszköz, a jelenségeket dinamikus mivoltukban képes bemutatni. A különböző természeti vagy akár társadalmi jelenségeket lelassítja vagy felgyorsítja, kicsinyíti vagy nagyítja úgy, hogy az esetleg láthatatlan és megfoghatatlan történést könnyen érthető modellé alakítja. Ez a modell nem veszti el a dinamikáját, nem kimerevített, statikus „fénykép" lesz, hanem megmaradnak jellemző tulajdonságai, hű marad az eredeti rendszerhez.

A számítógép és a modell megadja a változtatás lehetőségét. Az egyes jellemzők, paraméterek megváltoztatásával képesek vagyunk a való természet egy-egy kiválasztott jelenségét vizsgálni, sőt odáig is elmerészkedhetünk, hogy a modell viselkedését extrapolálva, a természetben elő nem forduló eseményeket is vizsgálatunkba vonhatunk.

Az élővilág jelenségei mind-mind időben változó, dinamikus folyamatok. Ez egyrészt igen alkalmassá teszi ezeket a jelenségeket a számítógépes modellezésre, másrészt ennek megvalósítása néha igen körülményes, esetleg lehetetlennek is látszik.

A biológiai történések számítógépes modelljei általában két, egymástól élesen el nem különíthető csoportba tartoznak. Az egyik a demonstrációs modellek csoportja, ahol egy közvetlenül nem észlelhető folyamatot mutatunk be a számítógép segítségével, kihasználva annak grafikai lehetőségeit. A másik csoportba a szimulációs modellek tartoznak, amelyeknél egyrészt lehetőség van az állandó beavatkozásra és a paraméterek befolyásolására, másrészt a számítógép — az esetleges demonstrációs értékű ábrák mellett — matematikailag is helyesen, például függvényekkel, grafikonokkal írja le a folyamatokat. Az első csoportba tartozhat például egy, a vérkörüket sematikusan ábrázoló rajz, ahol a vér útját követhetjük nyomon; a második csoportba tartozók közül példaként a populációdinamika Lotka—Volterra (nyúl-róka) modelljét említhetjük.

Augusztus 25—30. között Budapesten rendezték meg a Szocialista Országok Fővárosainak XIII. Ifjúsági Találkozóját. A találkozó rendező szerve a KISZ Budapesti Bizottsága volt.

A SZÁMALK Szakasits Árpád úti székházában rendezett politikai tanácskozáson négy földrész 14 szocialista és szocialista orientációjú országának képviseletében több mint másfélszáz fiatal vitatta meg, hogy az ifjúsági szövetségek mit tehetnek a fiatalok ideológiai, politikai neveléséért, szocialista meggyőződésük kialakításáért, a baráti országok közötti szolidaritás elmélyítéséért.

A SZOFIT-hagyományokhoz hiven a de-

A számítógép a biológia oktatásában

Általános problémák és lehetőségek

Közismert, hogy az iskolaszámítógép-program keretében az iskolákba kikerült számítógépekhez folyamatosan készülnek a felhasználásukat segítő oktatóprogramok. Természetesen először a matematika vagy fizika felől érkezők ismerkednek meg a gépekkel, és így e tantárgyakhoz ma már sokféle oktatóprogram kapható.

A biológiában közel sem ilyen jó a helyzet. Egyrészt a középiskolák többségében lévő HT-gépek igen gyenge grafikai lehetőségei talán éppen a biológia tanításában való felhasználást hátráltatják legjobban, hiszen erre a gépre csak olyan programok készülhetnek, melyek lemondanak a látványos szemléltetésről és csak igen egyszerű, sematikus ábrákat használnak. Másrészt a biológiai oktatóprogramok készítői között a legkevesebb a biológus, és még kevesebb a tanár. A programokat a biológiába elkalandozó matematikusok, fizikusok készítik, szinte kizárólag a biológia azon területeiről, melyek matematikailag elég jól megalapozottak (genetika, ökológia) és könnyen kezelhetők.

A hagyományos értelemben vett biológia mindebből teljesen kimarad. Sajnos, ennek hatására még magukban a biológiát tanítókban is felmerül a gondolat, hogy csak e szűk területen lehet használni a számítógépet a biológia oktatásában. Problémát jelent még az is, hogy pontosan e területek és e könnyen számítógépre vihető modellek (például életjáték) értelmezhetők nehezen a biológia adott szintjén, és így alkalmatlanok az alsó- vagy középfokú oktatásban való felhasználásra.

Nagy szükség lenne didaktikailag helyes, a tananyagba szervesen illeszkedő, szép kivitelű demonstrációs és tartalmas szimulációs programokra, de ezeknek születése körül még nincs minden rendben, és kevés a biztató kísérlet is.

Sajnos a biológusok és a biológiatanárok még nem barátkoztak meg eléggé a számítógépekkel, ami nem is nagy csoda addig, amíg sok helyen az iskolákban is nagyra nőtt számológépnek vélik a számítógépet, és így a matematika-, fizika- és technikatanárok fegyvertárába valónak tartják. Nincs olyan fórum, mely felhívná a figyelmet arra, hogy a nem számítástechnikai vagy matematikai alapképzettségű szakemberek is képesek használható oktatóprogramot írni. Lehet, hogy csak BASIC-ben, nehézkesebb, lassabb, esetleg néhány programozástechnikai hiányosságot tartalmazó programok jönnének létre, de e hátrányokat bőségesen kiegyenlítené a szaktárgyi és pedagógiai ismeret, hiszen végül is elsősorban a biológiatanár tudja, hogy mit és hogyan kellene, lehetne szimulálni és demonstrálni szaktárgyának oktatása során.

Mindezek alapján úgy gondolom, hogy oktatóprogram-pályázatokat kiírni nagyon kevés. Helyette egy jól felépített és meggondolt, egymásra épülő, tantárgyakra lebontott oktatási rendszerre lenne szükség, mely nagyjából a következő funkciókat lenne képes ellátni:

— Mindenekelőtt a már elkészült oktatóprogramok forgalmazásában lehetővé kellene tenni olyan bemutatókat, börzéket, ahol a leendő felhasználók részletesen megismerkedhetnek a programmal és esetleg a program alkalmazásának eddigi tapasztalataival.

— Az esetleg már meglévő BASIC nyelvű oktatóprogramokat el kellene látni olyan dokumentumentációval (például programlista), ami alapján a vállalkozó kedvűek — akik már járatosak kissé a programozásban — bátran saját céljaik szerint alakíthatnák azokat.

— A számítógép által még meg nem fertőzötteket — a potenciális felhasználókat — meg kell ismertetni a számítógéppel, és meg kell mutatni, hogy miként építhető be a számítógép az oktatásba.

— A különböző szaktárgyakat tanító tanároknak olyan BASIC-kurzusokat kellene szervezni, ahol a példaprogramok — már a legegyszerűbbek is — az adott szakterületről származnak, és így már az első órák után kézzelfogható közelségbe kerül az egyébként igen távoli cél, az önálló és a tanítási órán felhasználható program.

— Folyóiratokban, újságokban a programlisták közlése mellett arra is gondot kellene fordítani, hogy programötleteket kapjanak a tanárok, vagyis például a biológiában milyen jelenségeket lehetne egészen egyszerű módon szemléltetni és szimulálni, mert úgy tűnik, hogy a kevéssé egzakt tudományokban, tantárgyakban az egyik legnagyobb probléma éppen a modellezés alapját szolgáló jelenség kiválasztása.

Természetesen mindez csak általánosságban igaz. De azt hiszem, azzal mindenképpen egyet lehet érteni, hogy az iskolák felszerelése, gépekkel való ellátása mellett a mainál jóval nagyobb gondot kellene fordítani a gépek ésszerű felhasználására és a pedagógusok megnyerésére. Sajnos, a fejekben való rendcsinálás elég időigényes feladat, és ebben máris jelentős a lemaradásunk.

DEMETER LÁSZLÓ

Beszámoló a SZOFIT '86 programozói versenyéről

legációk tagjaiként több szakma képviselői is fővárosunkba érkeztek, hogy versenyeken mérjék össze tudásukat. Augusztus 27-én a női fodrászok, az autószerelők, az esztergályosok és a számítógép programozók vetélkedésére került sor.

A SZÁMALK a tanfolyami rendszerű oktatás bázisszervezőként kiemelkedő eredményeket ért el a számítástechnika oktatása terén is, ezért a verseny megszervezésére és lebonyolítására a SZÁMALK KISZ Bizottságát kérték fel a rendezők. A résztvevő fővárosok a következők voltak: Berlin, Budapest, Bukarest, Hanoi, Moszkva, Pnom Penh, Prága, Szófia, Varsó és Ulánbátor.

A külföldi fővárosok egy-egy versenyzővel vettek részt a versenyen. Budapestet két versenyző: Rákóczi Ferenc (MTA SZTAKI) és Szomor Attila (SZÁMALK) képviselte. Bukarest képviselője versenyen kívül oldotta meg a feladatokat. A versenyen a SZÁMALK Oktatási Irodáinak Commodore 16-os gépeit használták a résztvevők, akik hónapokkal korábban megkapták a versenykiírást, valamint a C16 BASIC-jének dokumentációját. A verseny előtti napon néhány órás gyakorlási lehetőséget is biztosítottunk a résztvevőknek. A versenyfeladatokat, melyeket alább közlünk, mindenki a saját anyanyelvén kapta kézhez.

A feladatok között egy 25 kérdéses teszt, adott program hibáinak felderítése, szétdarabolt program összerakása, különböző programozási problémák (például ébresztőóra-szimuláció, görberajzolás), valamint egy ismeretlen program működésének megfejtése szerepelt. (A szerkesztőségben a feladatok utólagus okulás céljából rendelkezésre állnak.)

Az ötórás „küzdelem" végén a zsűri eredményt hirdetett. Holtversenyben első helyezett lett Budapest (Rákóczi Ferenc) és Prága küldötte, második helyezést ért el Ulánbátor képviselője, a harmadik helyen pedig a varsói versenyző végzett.

SÜTŐ JÓZSEF PÁL

HT-1080Z 16/64

Nyomköve

Az új pályázatot követően remélhetőleg javulni fog az iskolák számítógéppel való ellátása. A régebbi HT gépek elérhetők lesznek olyan felhasználásra is, hogy a Z80 gépi kódú programozás iránt érdeklődő tanulók gyakorolhatnak velük. Ennek persze előfeltétele, hogy rendelkezésre álljanak azok a segédprogramok (EDI, EDTASM, TSAVE), amelyek sok szakkörben megtalálható, de — bár véleményem szerint csere útján való megszerzésük nem sért érdekeket — sok iskolában nem ismertek.

Az itt bemutatott program a programozás gyakorlását segíti. TRS—80 modell—III gépen fejlesztettem ki. Követelmény volt a kis méret (1 k a memória felső részén), a zárt utasításrendszer és a kényelmes kezelés. Segítségével futás közben lehet kipróbálni, megérteni saját vagy mások által készített programokat. Ellenőrizni lehet a memóriatartalmakat, módosítani a RAM értékeket, futás közben ellenőrizni változásukat. Akár lépésenként lehet látni vagy módosítani a processzor regisztereinek értékét.

Ez az eszköz természetesen csak segíti a munkát, de nem teszi lehetővé minden feladat megoldását.

A program működése

A programot a memória felső részébe töltjük be. Ezt a területet kell a bekapcsoláskor memóriavédelem alá helyeznünk. A memóriát kezelő parancsok saját munkaregisztereket használnak, és nem módosítják a BASIC-rendszerterületet.

A nyomkövetési parancsok az RST 30 (F7H) belépési ponton keresztül hívhatók. A program kívánt részén vagy elhelyezzük ezt az utasításkódot, vagy a töréspont parancsok kezelik beírva, majd minden bejelentkezéskor visszacserélve az eredeti tartalomra. A program minden bejelentkezése kicseréli a töréspontot eredeti tartalmára.

A program az itt közölt formájában kimenti a képernyő-memória alsó négy sorát, és továbbindításkor visszaírja. Így egy, a képernyőt is használó program működése jobban látható. A nyomkövető csak ezt az alsó négy sort használja. Nem menti ki viszont a kurzor (kiírás kezdete) értékét, de ez azért az ernyőn megjelenik. (Nem sikerült a programot tovább tömöríteni.) Ez a képernyőmentés további 256 bájt elfoglalását jelenti. Indokolt esetben 4 bájt 0-ra való átírásával ezt rövidre zárhatjuk.

Ez a működésmód a korlátokat is sejteti:

— csak RAM-ban lévő programok kipróbálására alkalmas,

— nem követhetők a hívások (RST8, CALL után elhelyezett operandus és a vermen keresztüli paraméterátadó program), ha a vermet indexelve használja,

— szubrutinhívások szólhatnak a fix memória- (ROM) területre, ilyenkor a töréspontot a hívás utáni visszatérésre tegyük.

A program két töréspontot kezel, így feltételes utasítások kényelmesebben kezelhetők vele.

A programhoz közlöm a teljes forrásnyelvi listát. Ez könnyebbé teszi a program megértését, és lehetővé teszi, ha valaki kedvet érez, saját ötlet, bővítés megbízhatóbb beépítését. A címkék egy részét célszerű saját címkére átírni, hogy a funkcióra utaljon.

Ismertetem a BASIC-ben futtatható töltőprogramot, amely hexadecimális alakban megadott DATA sorokból állítja elő a programot. Ugyancsak leírtam a DATA sorokat előállító BASIC-programot.

A használat során a már betöltött programot TSAVE-vel célszerűbb szalagra írni, és SYSTEM paranccsal beolvasni.

A betöltés

A különböző válaszként megadandó címeket a 64 k-s verzióra adom meg, zárójelbe írva a 16 k-s címeket.

Bekapcsoláskor a memória felső határa: 64535 (31767)

Ezután BASIC-ben írjuk be az *1. listán* közölt betöltőprogramot, és indítsuk el. (64 k esetén a 12. sort ki kell cserélni. A hozzá tartozó adatmező, amely a gépi kódú programot tartalmazza, 16 k esetén a *2. listán*, 64 k esetén a *3. listán* látható.) Lefutása után, ha hibátlanul írtuk be, a két kontrollösszeg megegyezik. Persze ettől még a program lehet hibás, ezért célszerű esetleges javítás céljából kazettára írni. A kontrollösszeg (12. sor): 121651 (108979).

Ha a betöltés sikerült (futásideje kb. 4 perc), az alábbi formában elindítható, és a nyomkövető az alsó négy sorban jelentkezik be.

```
Case
Memory Size? 64535
Radio Shack Model III Basic
(c)  80 Tandy
READY
>SYSTEM
```

D FC18 <CR> hatására megváltozik a kép:

```
      FC10:  F5 FD E5 CD 65 55 D5 7E   F5 21 00 00 39 11 00 FC
      FC20:  01 1A 00 ED B0 22 14 FC   F9 21 00 FB 21 00 3F 06
      FC30:  01 ED B0 2A 1A FC 2B 7E   FE F7 20 03 22 1E FC 21
Ce000FC40:  00 FC 06 02 4F 00 02 5E   77 23 5E 77 23 74 B3 28
```

Ezzel kezdhetjük a nyomkövetővel való ismerkedést.

A parancsok

— A (SCII) a memóriatartalom látható karakterek formájában jelenik meg.

```
FD60:   .  >  .  .  3  .  .  .  .  .  A  F  B  C  D
FD70:   E  H  L  I  X  1  Y  S  P  P  C  1  n  s  z  1
FD80:   h  1  p  n  c  .  .  .  .  .  .  .  (  .  "
FD90:   .  .  B  .  .  .  .  .  .  .  .  .  .  1  .
```

— H(exa) a kijelzés hexadecimális üzemmódba kerül.

Ha a memóriatartalmat átírjuk, ez a kijelzés változtatható. A többi parancsot célszerű egy kis program beírásával kezdeni, hogy a lehetőségeket lássuk. A program:

```
ORG   6000 H
CALL  6007 H
INC   HL
XOR   A
JR    6000 H
EX    DE, HL
RET
```

A beíráskor adjuk meg

D 6000 <CR>

M

és az értékeket folyamatosan írjuk.

```
      6000:  CD 07 60 23 AF 18 F9 EB   C9 00 00 CD C9 01 00 00
6004  6010:  00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
AF-   6020:  00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
      6030:  00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

Módosítani az érték beírásával lehet, az egy vagy két hexadecimális karaktert ,-vel zárva. Ilyenkor a következő címet írja ki, újabb beírást várva. "," növeli, "∇" csökkenti a címet.

A programbeírás ellenőrzése után az elindítást (először mindig)

G (o) Gaaa, b1, b2 <CR>

G, b1, b2 <CR>

G <CR>

paranccsal végezzük. aaaa az elindítás címe, b1; b2 a töréspont. Ha aaaa elmarad, a PC tartalma az indulási cím. Ha mindkét töréspontot elhagyjuk, nem kapjuk vissza a vezérlést, kilépünk a nyomkövetésből.

A töréspontón való megállás bejelentkezése:

program

```
AF   BC   DE   HL   IX   IY   SP   PC   in
1F44 0000 42EE FC1D 4015 4C46 428A 6043 19   -z---p--
     6000: CD 07 60 23 AF 18 F9 BE  C9 00 00 00 00 00 00 00
     6010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Az itt kiadható parancsok:— T(race) Állj meg újra itt, egy ciklust végrehajtva.

Ennél a parancsnál vigyázzunk, hogy ha nem kerül ide a vezérlés, elveszítjük az ellenőrzést!

— C(all) Lépésenként hajtja végre a programot, de CALL (RST) esetén csak a végrehajtás, majd visszatérés után jelentkezik újra.

```
AF   BC   DE   HL   IX   IY   SP   PC   in
0044 0000 423F 4184 4015 444C 41E5 6000 CD   -z---p--
     6000: CD 07 60 23 AF 18 F9 EB  C9 00 FE CD C9 01 00 00
     6010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

— I(nstruction) hatása azonos C-vel, de ha RAM szubrutinról van szó, ezt is lépésenként hajtja végre.

Természetesen a memóriaparancsok továbbra is minden bejelentkezéskor használhatóak.

— R(egiszter) R rp, vvvv <RC>
RP az AF, BC, DE, HL, IX, IY, SP, PC regiszterpár valamelyike, vvvv az új érték.

```
AF   BC   DE   HL   IX   IY   SP   PC   in
1F44 0000 42EE FC18 4015 0505 428A 600E 00   -z---p--
     6000: .CD.07 60 23 AF 18 F9 EB  C9 00 00 CD C9 01 00 00
RPC,6000
```

—; 64-gyel növeli a memóriatartalom-kiírás kezdőcímét.

— — 64-gyel csökkenti a kiírás kezdőcímét.

A módosító érték — bájt vagy szó — elönullákkal kerül értelmezésre, és az utolsó 2 vagy 4 hexa jegye értelmeződik. Továbbirással javítható, például 4041 bájt esetén 41, 9AA02 szó esetén AA02.

Használati tanácsok

A nyomkövetés menetét célszerű megtervezni. Ha saját programunkat próbáljuk, RST 30-at befordítva a program tetszőleges helyeire, itt *mindig* megáll a program, csak G xxxx paranccsal folytatható. Ha már nem szükséges, 00 (NOP) beírásával törölhetők ezek a pontok.

A segédprogramok

A kiíratő programot nem módosítottam, így a PEEK miatt csak 16 k-ban működik. Relokáltam a kiíratni kívánt programot a 16 k-s program helyére. A töltőben a POKE OR utasítással lett „becsapva", de ettől lassúbb lett. Így a 64 k-s memóriába is tölt.

Hogy a program kazettán is elérhető legyen, a Tudományszervezési és Informatikai Intézetnél megrendelhető a kazettán a 16 k-s vagy 64 k-s verzióban a BASIC-töltőprogram, az assembler forrásnyelv EDTASM formában és a binárisan, SYSTEM-mel tölthető verzióban. Ez egy 60 perces kazetta két oldalán van, az EDTASM programmal együtt.

Ha a programot SYSTEM paranccsal töltjük be (16 k esetén 31767)

```
>SYSTEM
*? 7E45E
```

>SYSTEM<CR> 16 k esetén /*?31768 CR-rel indítjuk el.
Ezután a már bemutatott parancsok használhatók.
Kilépés BASIC-be G 1DB4<CR>

Az ernyőmentés, visszamentés kiíratása

FC3B
cimen Ø, Ø
7C3B
FDBE
cimen Ø, Ø beírásával történik.
7DBE

A program szalagra mentéséhez
31768 — 32767-ig kezdőcím 31768
64536 — 65535-ig kezdőcím 64536
címeket kell megadni, hexa formában.
7C18, 7FFF, 7C18 a TSAVE programnak.
FC18, FFFF, FC18

A kommenteket nem fordítottam le, ezeket mindenki saját magának írja magyarázatként, és a megfejtéskor úgyis célszerű hozzáírni saját megjegyzésként.

A használható utasítások

A A memóriatartalmat ASCII (látható) formában írja ki. Ha vezérlő vagy grafikus karaktert kellene kiírni, *O*-t ír ki.

C Ha a lépésenkénti végrehajtás során ROM rutinhoz jutunk, a *C* parancs a töréspontot a CALL utáni pontra helyezi.

D Memóriatartalom kiírása 64 egymást követő cím tartalmának kiírásával (ASCII vagy hexa). Ha nem adunk címet, 32 karakteres megjelenítésből 64 karakteres kijelzésbe vált.

G Adott címen folytatás egy vagy két töréspont megadásával.

H Kijelzés átállítása hexa formátumra.

I Végrehajtás lépésenként.

M Memóriatartalom módosítása a D-ben megadott címtől kezdődően. "," növeli, "▽" csökkenti a módosítás címét.

R A processzor regisztereit módosíthatjuk.

T A megállás címén történő ismételt megállást tesz lehetővé.

Mielőtt adott program kipróbálására alkalmaznánk a programot, használatát be kell gyakorolni, hogy a megoldandó feladatra fordíthassuk figyelmünket!

SZOKOLY PÁL

```
1 CLEAR 10: ONERRORGOTO 0
2 DEFINT A-Z
3 C=5:CK#=0:V=0:C=0
4 READ B$:GOSUB 19:V=A3
5 READ B$:GOSUB 19:IF A3<128 THEN 6 ELSE C=-32768:A3=A3-128
6 V=A3#256+V:C=C+1
7 READ B$:IF B$="*" THEN 11
8 GOSUB 19:POKE V+C,A3:C=C+1:
9 CK#=CK#+A3
10 GOTO 7
11 CLS
12 A#=100979
13 PRINT"KONTROL-ÖSSZEG ="CK#" HELYESEN ="A#:PRINT
14 IF CK#=A# GOTO 17
15 PRINT"HIBÁS KONTROL-ÖSSZEG. CÉLSZERŰ A BEÍRÁST ELLENŐRIZNI."
16 END
17 PRINT"HIBÁTLAN KONTROL-ÖSSZEG. A PROGRAM BEÍRÁSA VALÓSZÍNŰEN SIKERÜLT."
18 END
19 H=ASC(MID$(B$,1,1)):L=ASC(MID$(B$,2,1))
20 IF H>57 THEN H=H-7
21 IF L>57 THEN L=L-7
22 A3=(H-48)*16+L-48:RETURN
```

1. lista. A BASIC-betöltőprogram

2. lista

```
23 DATA 18,7C,F5,FD,E5,DD,E5,E5,D5,C5,F5,21,00,00,39,11,00,7C,01,18,00,ED,B0,2B,
14,7C,F9,11,00,7B,21,00
24 DATA 3F,06,01,ED,B0,2A,16,7C,2B,7E,FE,F7,28,03,22,16,7C,21,00,7C,06,02,AF,0E,
02,5E,77,23,56,77,23,7A
25 DATA B3,28,07,1A,FE,F7,28,02,7E,12,23,10,E9,2A,FE,7F,7D,B4,28,0F,11,05,7C,CD,
D2,7D,21,00,00,22,FE,7F
26 DATA C9,9E,7D,ED,7B,14,7C,CD,E4,7C,21,C0,3F,22,20,40,CD,0E,7F,FE,47,CA,85,7D,
21,79,7C,E5,FE,3B,28,2E
27 DATA FE,2D,28,41,FE,41,28,42,FE,43,28,06,FE,48,28,3A,FE,49,CA,64,7E,FE,4D,CA,
E8,7D,FE,52,CA,22,7E,FE
28 DATA 54,CA,5C,7E,FE,44,C8,CD,AC,7F,28,0C,18,07,01,40,00,2A,96,7C,09,22,96,7C,
21,00,3F,06,04,22,20,40
29 DATA CD,51,7D,18,AB,01,C8,FF,18,E7,32,00,7C,C9,21,00,3F,22,20,40,CD,FA,7F,21,
6B,7D,06,09,7E,CD,33,00
30 DATA 23,7E,CD,33,00,23,CD,BA,7F,CD,BA,7F,CD,BA,7F,10,EB,CD,FA,7F,CD,F6,7F,21,
```

```
00,7C,E5,06,00,E3,3E,23
31 DATA 56,23,E5,EB,CD,DD,7F,CD,8A,7F,E1,E3,10,EF,2A,16,7C,CD,D9,7F,CD,8A,7F,CD,
8A,7F,2A,00,7C,4D,C5,21
32 DATA 7D,7D,06,06,CB,21,7E,38,02,3E,2D,CD,33,00,23,10,F3,C1,CD,FA,7F,E1,06,02,
CD,F6,7F,2A,06,7C,3E,C5
33 DATA CD,33,00,CD,DD,7F,CD,38,7F,10,05,3E,1D,C3,33,00,CD,F6,7F,18,E9,41,46,42,
43,44,45,48,4C,49,58,49
34 DATA 59,53,50,50,43,69,6E,73,7A,31,68,31,70,6E,63,06,02,11,05,7C,CD,AC,7F,28,
03,22,16,7C,38,0A,CD,AC
35 DATA 7F,F5,C4,D2,7D,F1,10,F4,21,18,7C,22,10,40,3E,C3,32,0F,40,21,15,7C,06,07,
56,28,5E,28,D5,10,F9,21
36 DATA 00,7B,11,00,3F,01,00,01,ED,B0,F1,C1,D1,E1,DD,E1,FD,E1,E1,F9,2A,16,7C,E5,
2A,0E,7C,C9,7E,12,18,3E
37 DATA F7,77,8E,C2,47,7C,7C,12,18,7D,12,18,C9,2A,06,7C,22,01,7C,E5,21,00,3F,22,
20,40,06,04,CD,51,7D,21
38 DATA 40,3F,22,20,40,2A,01,7C,CD,DD,7F,E5,21,80,3F,22,20,40,E1,CD,D9,7F,3E,2D,
CD,33,00,D1,CD,AC,7F,EB
39 DATA 28,01,73,D8,28,87,2E,C8,23,23,18,C4,CD,8E,7F,C8,4F,CD,8E,7F,C8,57,CD,8E,
7F,C0,D8,21,68,7D,06,00
40 DATA 7E,23,89,28,04,23,10,F8,C9,7E,BA,20,F8,3E,10,90,90,4F,06,00,21,08,7C,09,
E5,CD,FA,7F,D1,CD,AC,7F
41 DATA C8,EB,73,23,72,C9,2A,16,7C,22,FE,7F,3E,49,F5,ED,58,16,7C,1A,21,1C,7F,FE,
DD,28,0E,FE,FD,28,0A,21
42 DATA E7,7E,FE,ED,20,06,21,15,7F,13,1A,18,4F,7E,A1,23,BE,23,28,06,23,7E,FE,05,
30,F3,7E,47,E6,0F,6F,26
43 DATA 00,19,D5,11,05,7C,CD,D2,7D,E1,78,E6,F0,28,16,23,FE,20,38,35,28,27,FE,40,
38,17,28,0F,FE,60,38,08
44 DATA F1,FE,49,28,06,C3,A9,7D,2A,14,7C,7E,23,66,6F,18,18,4E,AF,CB,79,28,01,2F,
47,23,09,18,0F,2A,10,7C
45 DATA C8,69,28,08,2A,12,7C,18,03,2A,0E,7C,CD,D2,7D,18,D4,C7,C0,51,FF,C9,51,FF,
E9,11,CF,01,03,E7,22,03
46 DATA C7,C2,43,FF,C3,43,C7,C4,63,FF,CD,63,C7,06,02,F7,D3,02,C7,C6,02,FF,C8,02,
F7,10,32,E7,20,32,01,C7
47 DATA 43,04,F7,45,52,02,FE,34,03,C0,40,03,C0,80,03,FF,21,04,FF,22,04,FF,2A,04,
FF,36,04,FF,CB,04,FF,E9
48 DATA 22,02,C5,3E,3A,CD,33,00,CD,8A,7F,06,10,CD,68,7F,3A,00,7C,FE,41,20,13,CD,
8A,7F,7E,FE,20,38,04,FE
49 DATA C0,38,02,3E,2E,CD,33,00,23,AF,C4,D9,7F,10,DE,C1,AF,C9,ED,5B,01,7C,E5,B7,
ED,52,3E,95,28,0E,CD,86
50 DATA 7F,23,7D,84,E1,3E,AA,CA,33,00,18,08,CD,33,00,E1,78,FE,08,C0,3E,20,18,65,
D5,CD,49,00,D1,FE,0D,37
51 DATA C8,FE,20,38,07,28,C9,D5,CD,33,00,D1,FE,2C,C8,FE,41,3F,D0,E6,5F,C9,CD,8E,
7F,C8,21,00,00,CD,C8,7F
52 DATA DA,79,7C,29,29,29,29,B5,6F,CD,8E,7F,20,EF,1F,CE,01,C9,D6,30,D8,C6,E9,D8,
C6,06,38,03,C6,07,D8,C6
53 DATA 0A,87,C9,7E,23,18,05,7C,CD,E2,7F,7D,F5,1F,1F,1F,1F,CD,EB,7F,F1,E6,0F,C6,
90,27,CE,40,27,C3,33,00
54 DATA 3E,0D,18,F9,3E,1E,18,F5,00,00
55 DATA *
```

3. lista

```
23 DATA 18,FC,F5,FD,E5,DD,E5,E5,D5,C5,F5,21,00,00,39,11,08,FC,01,10,00,ED,B0,22,
14,FC,F9,11,00,FB,21,00
24 DATA 3F,06,01,ED,B0,2A,16,FC,2B,7E,FE,F7,20,03,22,16,FC,21,00,FC,06,02,AF,0E,
02,5E,77,23,56,77,23,7A
25 DATA B3,28,07,1A,FE,F7,20,02,7E,12,23,10,E9,2A,FE,FF,7D,B4,28,0F,11,05,FC,CD,
D2,FD,21,00,00,22,FE,FF
26 DATA C3,9E,FD,ED,7B,14,FC,CD,E4,FC,21,C0,3F,22,20,40,CD,8E,FF,FE,47,CA,85,FD,
21,79,FC,E5,FE,38,28,2E
27 DATA FE,2D,28,41,FE,41,28,42,FE,43,28,06,FE,48,28,3A,FE,49,CA,64,FE,FE,4D,CA,
E8,FD,FE,52,CA,22,FE,FE
28 DATA 54,CA,5C,FE,FE,44,C0,CD,AC,FF,28,0C,18,07,01,40,00,2A,06,FC,09,22,06,FC,
21,00,3F,06,04,B2,20,40
29 DATA CD,51,FD,18,AB,01,C0,FF,18,E7,32,00,FC,C9,21,00,3F,22,20,40,CD,FA,FF,21,
68,FD,06,00,7E,CD,33,00
30 DATA 23,7E,CD,33,00,23,CD,8A,FF,CD,8A,FF,CD,8A,FF,10,EB,CD,FA,FF,CD,F6,FF,21,
00,FC,E5,06,00,E3,5E,23
31 DATA 56,23,E5,EB,CD,DD,FF,CD,8A,FF,E1,E3,10,EF,2A,16,FC,CD,D9,FF,CD,8A,FF,CD,
8A,FF,2A,00,FC,4D,C5,21
32 DATA 7D,FD,06,08,CB,21,7E,38,02,3E,2D,CD,33,00,23,10,F3,C1,CD,FA,FF,E1,06,02,
CD,F6,FF,2A,06,FC,3E,C5
33 DATA CD,33,00,CD,DD,FF,CD,38,FF,10,05,3E,1D,C3,33,00,CD,F6,FF,18,E9,41,46,42,
43,44,45,48,4C,49,58,49
34 DATA 59,53,50,50,43,69,6E,73,7A,31,68,31,70,6E,63,06,02,11,05,FC,CD,AC,FF,28,
03,22,16,FC,38,0A,CD,AC
35 DATA FF,F5,C4,D2,FD,F1,10,F4,21,18,FC,22,10,40,3E,C3,32,0F,40,21,15,FC,06,07,
56,28,5E,28,D5,10,F9,21
36 DATA 00,FB,11,00,3F,01,00,01,ED,B0,F1,C1,D1,E1,DD,E1,FD,E1,E1,F9,2A,16,FC,E5,
2A,0E,FC,C9,7E,12,18,3E
37 DATA F7,77,8E,C2,47,FC,7C,12,18,7D,12,18,C9,2A,06,FC,22,01,FC,E5,21,00,3F,22,
20,40,06,04,CD,51,FD,21
38 DATA 40,3F,22,20,40,2A,01,FC,CD,DD,FF,E5,21,80,3F,22,20,40,E1,CD,D9,FF,3E,2D,
CD,33,00,D1,CD,AC,FF,EB
39 DATA 28,01,73,D8,28,87,28,C8,23,23,18,C4,CD,8E,FF,C8,4F,CD,8E,FF,C8,57,CD,8E,
FF,C0,D8,21,68,FD,06,00
40 DATA 7E,23,89,28,04,23,10,F8,C9,7E,BA,20,F8,3E,10,90,90,4F,06,00,21,08,FC,09,
E5,CD,FA,FF,D1,CD,AC,FF
41 DATA C8,EB,73,23,72,C9,2A,16,FC,22,FE,FF,3E,49,F5,ED,5B,16,FC,1A,21,1C,FF,FE,
DD,28,0E,FE,FD,28,0A,21
42 DATA E7,FE,FE,ED,20,06,21,15,FF,13,1A,18,4F,7E,A1,23,BE,23,28,06,23,7E,FE,05,
30,F3,7E,47,E6,0F,6F,26
43 DATA 00,19,D5,11,05,FC,CD,D2,FD,E1,78,E6,F0,28,16,23,FE,20,38,35,28,27,FE,40,
38,17,28,0F,FE,60,38,08
44 DATA F1,FE,49,28,06,C3,A9,FD,2A,14,FC,7E,23,66,6F,18,18,4E,AF,CB,79,28,01,2F,
47,23,09,18,0F,2A,10,FC
45 DATA C8,69,28,08,2A,12,FC,18,03,2A,0E,FC,CD,D2,FD,18,D4,C7,C0,51,FF,C9,51,FF,
E9,11,CF,01,03,E7,22,03
46 DATA C7,C2,43,FF,C3,43,C7,C4,63,FF,CD,63,C7,06,02,F7,D3,02,C7,C6,02,FF,C8,02,
F7,10,32,E7,20,32,01,C7
47 DATA 43,04,F7,45,52,02,FE,34,03,C0,40,03,C0,80,03,FF,21,04,FF,22,04,FF,2A,04,
FF,36,04,FF,CB,04,FF,E9
48 DATA 22,02,C5,3E,3A,CD,33,00,CD,8A,FF,06,10,CD,68,FF,3A,00,FC,FE,41,20,13,CD,
8A,FF,7E,FE,20,38,04,FE
49 DATA C0,38,02,3E,2E,CD,33,00,23,AF,C4,D9,FF,10,DE,C1,AF,C9,ED,5B,01,FC,E5,B7,
ED,52,3E,95,28,0E,CD,86
50 DATA FF,23,7D,84,E1,3E,AA,CA,33,00,18,08,CD,33,00,E1,78,FE,08,C0,3E,20,18,65,
D5,CD,49,00,D1,FE,0D,37
51 DATA C8,FE,20,38,07,28,C9,D5,CD,33,00,D1,FE,2C,C8,FE,41,3F,D0,E6,5F,C9,CD,8E,
FF,C8,21,00,00,CD,C8,FF
52 DATA DA,79,FC,29,29,29,29,B5,6F,CD,8E,FF,20,EF,1F,CE,01,C9,D6,30,D8,C6,E9,D8,
C6,06,38,03,C6,07,D8,C6
53 DATA 0A,87,C9,7E,23,18,05,7C,CD,E2,FF,7D,F5,1F,1F,1F,1F,CD,EB,FF,F1,E6,0F,C6,
90,27,CE,40,27,C3,33,00
54 DATA 3E,0D,18,F9,3E,1E,18,F5,00,00
55 DATA *
```

4. lista. A program forrásnyelvi listája 16 k-ra fordítva

```
                ;          HT-1080 lepeskoveto 16-K
                ;
                ; Parancsok      Dxxxx           kiir 64 byte-ot
                ;                ;               cimet 64-el noveli
                ;                -               cimet 64-el csokkenti
                ;                I               lepesenkent kovet
                ;                C               call-nal a ret utan kovet
                ;                M               memoria modositasa
                ;                                mod.cim 1-el novelese
                ;                - -             mod.cim csokkentese 1-el
                ;                                yy byte modositasa
                ;                                barmi mas lezarja a mod.
                ;
                ;                Sxxxx,B1,B2     xxxx-tol B1,B2 torespont
                ;                                betevesevel folytatas
                ;                T               a PC jelenlegi erteken
                ;                                alljon meg legkozelebb
                ;                A               a tartalom ASCII-formaban
                ;                H               HEXA-kiirasi forma
                ;                barmi mas       Register+32 byte-os forma
                ;
                ;  af bc de hl ix iy sp pc in    flagek, CPU-regiszterek
                ;                                kijelzese (32 byte nem)
                ;
                ;                Rrr,xxxx        beir a CPU regiszterekbe
                ;                                AF BC DE HL IX IY SP PC
                ;
                ; Menti az ernyokepet, a cursort NEM !
                ;
                ; Exit    G1D8A<ENTER>
                ;
7B00                    ORG     07B00H
3F00            ACF     EQU     3F00H    ; 0.
3F40            ACG     EQU     3F40H    ; 1.
3F80            ACH     EQU     3F80H    ; 2.
3FC0            ACI     EQU     3FC0H    ; 3. sor
4020            CUR     EQU     4020H    ; cursor pozicio
                ;
0100            SBUF:   DEFS    256      ; Ernyo buffer
0001            ASC:    DEFS    1        ; Ascii/Hexa
0002            ACN:    DEFS    2        ; torespont tarolasa
0002                    DEFS    2
0001            ACO:    DEFS    1
0002            ACP:    DEFS    2        ; Az ernyo pointer cime
                ;
0006            REG:    DEFS    6        ; AF-BC-DE
0002            RHL:    DEFS    2        ; HL
0002            RIX:    DEFS    2        ; IX
0002            RIY:    DEFS    2        ; IY
0001            RSP:    DEFS    1        ; SP
0001            ACV:    DEFS    1
0002            RPC:    DEFS    2        ; PC
                ;
                ;
                ; 1986 05 01  Trace, ernyo mentessel, 'T' paranccsal
                ;
7C18 F5         STRT:   PUSH    AF               ; CPU regiszterek mentese
7C19 FDE5               PUSH    IY
7C1B DDE5               PUSH    IX
7C1D E5                 PUSH    HL
7C1E D5                 PUSH    DE
7C1F C5                 PUSH    BC
7C20 F5                 PUSH    AF
7C21 210000             LD      HL,0
7C24 39                 ADD     HL,SP
7C25 11087C             LD      DE,REG
7C28 011000             LD      BC,10H
7C2B EDB0               LDIR                     ; stack atmasolasa
7C2D 22147C             LD      (RSP),HL
7C30 F9                 LD      SP,HL
7C31 11007B             LD      DE,SBUF
7C34 21003F             LD      HL,ACF
7C37 0601               LD      B,1              ; 256
7C39 EDB0               LDIR                     ; ernyo atirasa
7C3B 2A167C             LD      HL,(RPC)
7C3E 2B                 DEC     HL
7C3F 7E                 LD      A,(HL)
7C40 FEF7               CP      0F7H
7C42 2003               JR      NZ,ADB
7C44 22167C             LD      (RPC),HL         ; torespont vissza
7C47 21007C     ADB:    LD      HL,ASC           ; cserelese
7C4A 0602               LD      B,2
7C4C AF         ADC:    XOR     A
7C4D 0E02               LD      C,2
7C4F 5E                 LD      E,(HL)
7C50 77                 LD      (HL),A
7C51 23                 INC     HL
7C52 56                 LD      D,(HL)
7C53 77                 LD      (HL),A
7C54 23                 INC     HL
7C55 7A                 LD      A,D
7C56 B3                 OR      E
7C57 2807               JR      Z,ADD
7C59 1A                 LD      A,(DE)
7C5A FEF7               CP      0F7H
7C5C 2002               JR      NZ,ADD
7C5E 7E                 LD      A,(HL)
7C5F 12                 LD      (DE),A
7C60 23         ADD:    INC     HL
7C61 10E9               DJNZ    ADC
7C63 2AFE7F             LD      HL,(RPCE)
7C66 7D                 LD      A,L
7C67 B4                 OR      H
7C68 280F               JR      Z,ADE            ; nem 'T'
7C6A 11057C             LD      DE,ACO
7C6D CD027D             CALL    BRK              ; ha T torespont betevese
7C70 210000             LD      HL,0
7C73 22FE7F             LD      (RPCE),HL        ; 'T' pointer torlese
7C76 C39E7D             JP      AEF
                ;
7C79 ED7B147C   ADE:    LD      SP,(RSP)
7C7D CDE47C             CALL    ADP
7C80 21C03F             LD      HL,ACI
7C83 222040             LD      (CUR),HL
7C86 CD0E7F     ADA:    CALL    CIN              ; read parancs
7C89 FE47               CP      'G'
```

```
7C8B CA[illegible]               JP    Z,GOD
7C8E 21[illegible]               LD    HL,ADE
7C91 E5                          PUSH  HL
7C92 FE3B                        CP    ';'
7C94 282E                        JR    Z,ADJ
7C96 FE2D                        CP    '-'
7C98 2841                        JR    Z,ADN
7C9A FE41                        CP    'A'
7C9C 2842                        JR    Z,ADD
7C9E FE43                        CP    'C'
7CA0 2806                        JR    Z,ADF
7CA2 FE48                        CP    'H'
7CA4 283A                        JR    Z,ADO
7CA6 FE49                        CP    'I'
7CA8 CA647E            ADF:      JP    Z,STP
7CAB FE4D                        CP    'M'
7CAD CAE37D                      JP    Z,MOD
7CB0 FE52                        CP    'R'
7CB2 CA227E                      JP    Z,RMD
7CB5 FE54                        CP    'T'
7CB7 CA5C7E                      JP    Z,ATW
7CBA FE44                        CP    'D'
7CBC C0                          RET   NZ
7CBD CDAC7F                      CALL  AFY
7CC0 280C                        JR    Z,AHX
7CC2 1807                        JR    AHY
                       ;
7CC4 014000            ADJ:      LD    BC,40H          ; 64-el novel
7CC7 2A067C            ADK:      LD    HL,(ACP)
7CCA 09                          ADD   HL,BC
7CCB 22067C            AHY:      LD    (ACP),HL
7CCE 21003F            AHX:      LD    HL,ACF
7CD1 0604                        LD    B,4
7CD3 222040                      LD    (CUR),HL
7CD6 CD517D                      CALL  ADW
7CD9 18A8                        JR    ADA
                       ;
7CDB 01C0FF            ADN:      LD    BC,0FFC0H       ; 64-el csokkent
7CDE 18E7                        JR    ADK
                       ;
7CE0 32007C            ADO:      LD    (ABC),A
7CE3 C9                          RET
                       ;
7CE4 21003F            ADP:      LD    HL,ACF
7CE7 222040                      LD    (CUR),HL        ; cursor pozicionalasa
7CEA CDFA7F                      CALL  AGJ
7CED 21687D                      LD    HL,HDR
7CF0 0609                        LD    B,9
7CF2 7E                AOQ:      LD    A,(HL)
7CF3 CD3300                      CALL  33H
7CF6 23                          INC   HL
7CF7 7E                          LD    A,(HL)
7CF8 CD3300                      CALL  33H
7CFB 23                          INC   HL
7CFC CD8A7F                      CALL  AGH
7CFF CD8A7F                      CALL  AGH
7D02 CD8A7F                      CALL  AGH
7D05 10EB                        DJNZ  AOQ
7D07 CDFA7F                      CALL  AGJ
7D0A CDF67F                      CALL  CRF
7D0D 21087C                      LD    HL,REG
7D10 E5                          PUSH  HL
7D11 0608                        LD    B,8
7D13 E3                AAE:      EX    (SP),HL
7D14 5E                          LD    E,(HL)
7D15 23                          INC   HL
7D16 56                          LD    D,(HL)
7D17 23                          INC   HL
7D18 E5                          PUSH  HL
7D19 EB                          EX    DE,HL
7D1A CD0D7F                      CALL  HXW
7D1D CD8A7F                      CALL  AGH
7D20 E1                          POP   HL
7D21 E3                          EX    (SP),HL
7D22 10EF                        DJNZ  AAE
7D24 2A167C                      LD    HL,(RPC)
7D27 CD097F                      CALL  AGC
7D2A CD8A7F                      CALL  AGH
7D2D CD8A7F                      CALL  AGH
7D30 2A087C                      LD    HL,(REG)
7D33 4D                          LD    C,L
7D34 C5                          PUSH  BC
7D35 21707D                      LD    HL,FLG
7D38 0608                        LD    B,8
7D3A CB21              ADR:      SLA   C
7D3C 7E                          LD    A,(HL)
7D3D 3802                        JR    C,ADS
7D3F 3E20                        LD    A,' '
7D41 CD3300            ADS:      CALL  33H
7D44 23                          INC   HL
7D45 10F3                        DJNZ  ADR
7D47 C1                          POP   BC
7D48 CDFA7F                      CALL  AGJ
7D4B E1                          POP   HL
7D4C 0602                        LD    B,2
7D4E CDF67F                      CALL  CRF
7D51 2A067C            ADW:      LD    HL,(ACP)
7D54 3EC5              ADX:      LD    A,0C5H
7D56 CD3300                      CALL  33H
7D59 CD0D7F                      CALL  HXW
7D5C CD397F                      CALL  AFN
7D5F 1005                        DJNZ  ADV
7D61 3E1D                        LD    A,1DH
7D63 C33300                      JP    33H
                       ;
7D66 CDF67F            ADV:      CALL  CRF
7D69 18E9                        JR    ADX
                       ;
7D6B 41                HDR:      DEFM  'AFBCDEHL'      ; Fejlec regiszterekhez
7D73 49                          DEFM  'IXIYSPPCin'
7D7D 73                FLG:      DEFM  'szhipnc'
                       ;
7D85 0602              GOD:      LD    B,2             ; G
7D87 11657C                      LD    DE,ACO
7D8A CDAC7F                      CALL  AFY
7D8D 2803                        JR    Z,AED
```

```
7D8F 22167C                      LD    (RPC),HL
7D92 380A              AED:      JR    C,AEF
7D94 CDAC7F                      CALL  AFY
7D97 F5                          PUSH  AF
7D98 C4D27D                      CALL  NZ,BRK
7D9B F1                          POP   AF
7D9C 10F4                        DJNZ  AED
7D9E 21187C            AEF:      LD    HL,STRT         ; Belepes nyomkovetesbe
7DA1 221040                      LD    (4010H),HL
7DA4 3EC3                        LD    A,0C3H
7DA6 320F40                      LD    (400FH),A
7DA9 21157C            AEH:      LD    HL,ACV
7DAC 0607                        LD    B,7
7DAE 56                AEJ:      LD    D,(HL)          ; Torespontok betevese
7DAF 2B                          DEC   HL
7DB0 5E                          LD    E,(HL)
7DB1 2B                          DEC   HL
7DB2 D5                          PUSH  DE
7DB3 10F9                        DJNZ  AEJ
7DB5 21007B                      LD    HL,EBUF
7DB8 11003F                      LD    DE,ACF
7DBB 010001                      LD    BC,100H
7DBE EDB0                        LDIR                  ; Ernyo visszamentese
7DC0 F1                          POP   AF              ; Regiszterek visszairasa
7DC1 C1                          POP   BC
7DC2 D1                          POP   DE
7DC3 E1                          POP   HL
7DC4 DDE1                        POP   IX
7DC6 FDE1                        POP   IY
7DC8 E1                          POP   HL
7DC9 F9                          LD    SP,HL
7DCA 2A1E7C                      LD    HL,(RPC)
7DCD E5                          PUSH  HL
7DCE 2A0E7C                      LD    HL,(RHL)
7DD1 C9                          RET                   ; exit
                       ;
7DD2 7E                BRK:      LD    A,(HL)          ; Torespont kezelese
7DD3 12                          LD    (DE),A
7DD4 1B                          DEC   DE
7DD5 3EF7                        LD    A,0F7H
7DD7 77                          LD    (HL),A
7DD8 BE                          CP    (HL)
7DD9 C2477C                      JP    NZ,ADB
7DDC 7C                          LD    A,H
7DDD 12                          LD    (DE),A
7DDE 1B                          DEC   DE
7DDF 7D                          LD    A,L
7DE0 12                          LD    (DE),A
7DE1 1B                          DEC   DE
7DE2 C9                          RET
                       ;
7DE3 2A067C            MOD:      LD    HL,(ACP)        ; Memoria modositasa
7DE6 22017C            AEP:      LD    (ACN),HL
7DE9 E5                          PUSH  HL
7DEA 21003F                      LD    HL,ACF
7DED 222040                      LD    (CUR),HL
7DF0 0604                        LD    B,4
7DF2 CD517D                      CALL  ADW
7DF5 21403F                      LD    HL,ACG
7DF8 222040                      LD    (CUR),HL
7DFB 2A017C                      LD    HL,(ACN)
7DFE CD0D7F                      CALL  HXW
7E01 E5                          PUSH  HL
7E02 21803F                      LD    HL,ACH
7E05 222040                      LD    (CUR),HL
7E08 E1                          POP   HL
7E09 CD097F                      CALL  AGC
7E0C 3E2D                        LD    A,'-'
7E0E CD3300                      CALL  33H
7E11 D1                          POP   DE
7E12 CDAC7F                      CALL  AFY
7E15 EB                          EX    DE,HL
7E16 2801                        JR    Z,AEQ
7E18 73                          LD    (HL),E
7E19 D8                AEQ:      RET   C
7E1A 2B                          DEC   HL
7E1B B7                          OR    A
7E1C 28C8                        JR    Z,AEP
7E1E 23                          INC   HL
7E1F 23                          INC   HL
7E20 18C4                        JR    AEP
                       ;
7E22 CD9E7F            RMD:      CALL  CIN             ; Regiszter modositasa
7E25 C8                          RET   Z
7E26 4F                          LD    C,A
7E27 CD9E7F                      CALL  CIN
7E2A C8                          RET   Z
7E2B 57                          LD    D,A
7E2C CD9E7F                      CALL  CIN
7E2F C0                          RET   NZ
7E30 D8                          RET   C
7E31 21687D                      LD    HL,HDR
7E34 0608                        LD    B,8
7E36 7E                AET:      LD    A,(HL)
7E37 23                          INC   HL
7E38 B9                          CP    C
7E39 2804                        JR    Z,AEW
7E3B 23                AEU:      INC   HL
7E3C 10F8                        DJNZ  AET
7E3E C9                          RET
                       ;
7E3F 7E                AEW:      LD    A,(HL)
7E40 BA                          CP    D
7E41 20F8                        JR    NZ,AEU
7E43 3E10                        LD    A,10H
7E45 90                          SUB   B
7E46 90                          SUB   B
7E47 4F                          LD    C,A
7E48 0600                        LD    B,0
7E4A 21087C                      LD    HL,REG
7E4D 09                          ADD   HL,BC
7E4E E5                          PUSH  HL
7E4F CDFA7F                      CALL  AGJ
7E52 D1                          POP   DE
7E53 CDAC7F                      CALL  AFY
7E56 C8                          RET   Z
```

```
7ED1 EB        00349        EX    DE,HL
7ED2 73        00350        LD    (HL),E
7ED3 23        00351        INC   HL
7ED4 72        00352        LD    (HL),D
7ED5 C9        00353        RET
               00354 ;
7ED6 2A187C    00355 AFH:   LD    HL,(RFC)
7ED9 22F67F    00356        LD    (RPIE),HL      ; Elozo ertek
7EDC 3E49      00357        LD    A, 'I'
               00358 ;
7EDE F5        00359 SIP:   PUSH  AF               ; I,C lenesenkint
7EDF ED5B187C  00360        LD    DE,(RFC)
7EE3 1A        00361        LD    A,(DE)
7EE4 6117FF    00362        LD    HL,UT2
7EE7 F61D      00363        OR    1DH
7EE9 FE4D      00364        CP    4DH            ; IX tipusu
7EEB 2808      00365        JR    Z,AE1
7EED FE5D      00366        CP    5DH
7EEF 2804      00367        JR    Z,AE1          ; IY tipusu
7EF1 2E7F      00368        LD    L,UT1
7EF3 FEED      00369        CP    0EDH           ; ED tipusu
7EF5 2808      00370        JR    NZ,AE2
7EF7 2115FF    00371        LD    HL,UT2
7EFA 13        00372 AE1:   INC   DE
7EFB 1A        00373        LD    A,(DE)
7EFC 1B        00374        DEC   DE
7EFD 4F        00375 AE2:   LD    C,A
7EFE 7E        00376 AFA:   LD    A,(HL)
7EFF A1        00377        AND   C
7F00 22        00378        INC   HL
7F01 BE        00379        CP    (HL)
7F02 23        00380        INC   HL
7F03 2006      00381        JR    Z,AFB
7F05 23        00382        INC   HL
7F06 7E        00383        LD    A,(HL)
7F07 FE45      00384        CP    45
7F09 3802      00385        JR    NC,AFA
7F0B 7E        00386 AFB:   LD    A,(HL)
7F0C 47        00387        LD    B,A
7F0D E60F      00388        AND   0FH
7F0F 6F        00389        LD    L,A
7F10 2600      00390        LD    H,0
7F12 19        00391        ADD   HL,DE
7F13 D5        00392        PUSH  DE
7F14 114D7C    00393        LD    DE,ACO
7F17 CD027D    00394        CALL  BRK
7F1A E1        00395        POP   HL
7F1B 78        00396        LD    A,B
7F1C E60F      00397        AND   0FH
7F1E 2816      00398        JR    Z,AFC
7F20 23        00399        INC   HL
7F21 FE09      00400        CP    09H
7F23 3805      00401        JR    C,AFI
7F25 2807      00402        JR    Z,AFH
7F27 FE40      00403        CP    40H
7F29 3817      00404        JR    C,AFF
7F2B 2807      00405        JR    Z,AFE
7F2D FE46      00406        CP    46H
7F2F 3809      00407        JR    C,AFD
7F31 F1        00408        POP   AF
7F32 FE49      00409        CP    'I'
7F34 2806      00410        JR    Z,AFE
7F36 C39D7D    00411 AFC:   JP    AEH
               00412 ;
7F39 3A147C    00413 AFD:   LD    HL,(REP)
7F3C 7E        00414 AFE:   LD    A,(HL)
7F3D 23        00415        INC   HL
7F3E 66        00416        LD    H,(HL)
7F3F 6F        00417        LD    L,A
7F40 1818      00418        JR    AFJ
               00419 ;
7F42 4E        00420 AFF:   LD    C,(HL)
7F43 AF        00421        XOR   A
7F44 CB79      00422        BIT   7H,C
7F46 2001      00423        JR    Z,AFG
7F48 2F        00424        CPL
7F49 47        00425 AFG:   LD    B,A
7F4A 23        00426        INC   HL
7F4B 66        00427        LD    H,(HL)
7F4C 6F        00428        LD    L,A
7F4D 1801      00429        JR    AFJ
               00430 ;
7F4F 2A187C    00431 AFH:   LD    HL,(RIX)
7F52 CB69      00432        BIT   5H,C
7F54 2800      00433        JR    Z,AFJ
7F56 2A187C    00434        LD    HL,(RIY)
7F59 1803      00435        JR    AFJ
               00436 ;
7F5B 2A8E7C    00437 AFI:   LD    HL,(RHL)
7F5E CD027D    00438 AFJ:   CALL  BRK
7F61 1804      00439        JR    AFC
               00440 ;
7F63 C7C8      00441 JT1:   DEFW  0C8C7H           ; feltetelles JP
7F65 51        00442        DEFB  51H
7F66 FFC9      00443        DEFW  0C9FFH           ; ret
7F68 51        00444        DEFB  51H
7F69 FFE9      00445        DEFW  0E9FFH           ; JP (hl)
7F6B 11        00446        DEFB  11H
7F6C CF01      00447        DEFW  01CFH
7F6E 03        00448        DEFB  3
7F6F E7ED      00449        DEFW  0ED E7H
7F71 03        00450        DEFB  3
7F72 C7C2      00451        DEFW  0C2C7H           ; JP cc
7F74 43        00452        DEFB  43H
7F75 FFC3      00453        DEFW  0C3FFH           ; JP xxxx
7F77 43        00454        DEFB  43H
7F78 C7C4      00455        DEFW  0C4C7H           ; CALL cc
7F7A 63        00456        DEFB  63H
7F7B FFCD      00457        DEFW  0CDFFH           ; Call xxxx
7F7D 63        00458        DEFB  63H
7F7E C79E      00459        DEFW  9EC7H
7F80 02        00460        DEFB  2
7F81 F7DB      00461        DEFW  0DBF7H           ; In/Out
7F83 02        00462        DEFB  2
7F84 C7C6      00463        DEFW  0C6C7H
7F86 02        00464        DEFB  2
7F87 FFCB      00465        DEFW  0CBFFH
7F89 02        00466        DEFB  2
```

```
7F8B F7C6      00465        DEFW  10F7H            ; Djnz xx
7F8D 25        00466        DEFB  25H
7F8E E720      00467        DEFW  20E7H            ; JR cc
7F90 32        00468        DEFB  32H
7F91 91        00469        DEFB  '
7F92 C742      00470 UT2:   DEFW  43C7H            ; Ld (xx),Reg
7F94 94        00471        DEFB  4
7F95 F745      00472        DEFW  45F7H            ; Retn,Reti
7F97 52        00473        DEFB  52H
7F98 02        00474        DEFB  2
               00475 ;
7F9A FE34      00476 UT3:   DEFW  34FEH
7F9C 03        00477        DEFB  3
7F9D C040      00478        DEFW  40C0H            ; Ld Reg,(xx)
7F9F 03        00479        DEFB  3
7FA0 7865      00480        DEFW  6578H

7FA2 02        00481        DEFB  3
7FA3 FF21      00482        DEFW  21FFH
7FA5 04        00483        DEFB  4
7FA6 FF22      00484        DEFW  22FFH
7FA8 04        00485        DEFB  4
7FA9 FF2A      00486        DEFW  2AFFH
7FAB 04        00487        DEFB  4
7FAC FF36      00488        DEFW  36FFH
7FAE 04        00489        DEFB  4
7FAF FFCB      00490        DEFW  0CBFFH
7FB1 04        00491        DEFB  4
7FB2 FF29      00492        DEFW  0E9FFH           ; JP (ix,iy)
7FB4 22        00493        DEFB  22H
7FB5 02        00494        DEFB  2
               00495 ;
7FB6 C5        00496 AFN:   PUSH  BC               ; ASCII kiiras
7FB7 3E3A      00497        LD    A,':'
7FB9 CD3300    00498        CALL  33H
7FBC CDBA7F    00499        CALL  AGH
7FBF 0610      00500        LD    B,1E
7FC1 CD687F    00501 AFO:   CALL  AF1
7FC4 3A007C    00502        LD    A,(ASC)
7FC7 FE41      00503        CP    'A'
7FC9 2013      00504        JR    NZ,AFS
7FCB CDBA7F    00505        CALL  AGH
7FCE 7E        00506        LD    A,(HL)
7FCF FE06      00507        CP    6
7FD1 3004      00508        JR    C,AFQ
7FD3 FEC8      00509        CP    0C8H
7FD5 3002      00510        JR    C,AFR
7FD7 3E2E      00511 AFQ:   LD    A,'.'
7FD9 CD3300    00512 AFR:   CALL  33H
7FDC 23        00513        INC   HL
7FDD AF        00514        XOR   A
7FDE CD977F    00515 AFS:   CALL  AG2C
7FE1 10DE      00516        DJNZ  AFO
7FE3 C1        00517        POP   BC
7FE4 AF        00518 AAG:   XOR   A
7FE5 C9        00519        RET
               00520 ;
7FE6 ED5B017C  00521 AFT:   LD    DE,(ACN)
7FEA E5        00522        PUSH  HL
7FEB 37        00523        SCF
7FEC ED52      00524        SBC   HL,DE
7FEE 3E95      00525        LD    A,95H
7FF0 280E      00526        JR    Z,AFU            ; marker betevese
7FF2 CDB67F    00527        CALL  AFV
7FF5 23        00528        INC   HL
7FF6 7D        00529        LD    A,L
7FF7 B4        00530        OR    H
7FF8 E1        00531        POP   HL
7FF9 3EAA      00532        LD    A,0AAH
7FFB CA3300    00533        JP    Z,33H            ; marker betevese
7FFE 1808      00534        JR    AGH
               00535 ;
8000 CD3300    00536 AFU:   CALL  33H
8003 E1        00537        POP   HL
8004 78        00538 AFV:   LD    A,B
8005 FE08      00539        CP    8
8007 C0        00540        RET   NZ
8008 3E20      00541 AGH:   LD    A,' '            ; utolso byte a sorban
800A 1865      00542        JR    AGG
               00543 ;
800C D5        00544 CIN:   PUSH  DE
800D CD4900    00545        CALL  49H              ; konzol input
8010 D1        00546        POP   DE
8011 FE0D      00547        CP    0DH
8013 37        00548        SCF
8014 C8        00549        RET   Z
8015 FE20      00550        CP    ' '
8017 3807      00551        JR    C,AAF
8019 28C9      00552        JR    Z,AAG
801B D5        00553        PUSH  DE
801C CD3300    00554        CALL  33H
801F D1        00555        POP   DE
8020 FE2C      00556 AAF:   CP    ','
8022 C9        00557        RET   Z
8023 FE41      00558        CP    'A'
8025 3F        00559        CCF
8026 D8        00560        RET   NC
8027 E65F      00561        AND   5FH
8029 C9        00562        RET
               00563 ;
802A CDBE7F    00564 AFY:   CALL  CIN
802D C8        00565        RET   Z
802E 214000    00566        LD    HL,0
8031 CDC67F    00567 AF2:   CALL  HX1
8034 DA797C    00568        JP    C,ADE
8037 29        00569        ADD   HL,HL
8038 29        00570        ADD   HL,HL
8039 29        00571        ADD   HL,HL
803A 29        00572        ADD   HL,HL
803B B5        00573        OR    L
803C 6F        00574        LD    L,A
803D CDBE7F    00575        CALL  CIN
8040 28EF      00576        JR    NZ,AF2
8042 1F        00577        RRA
8043 CE81      00578        ADC   A,81H
8045 C9        00579        RET
```

```
                00580 ;
7FCB D630       00581 HXI:      SUB     '0'             ; Hexa beolvasas
7FCA D8         00582           RET     C
7FCB CEE9       00583           ADD     A,0E9H
7FCD D8         00584           RET     C
7FCE C606       00585           ADD     A,6
7FD0 3803       00586           JR      C,AGB
7FD2 C607       00587           ADD     A,7
7FD4 D8         00588           RET     C
7FD5 C60A       00589 AGB:      ADD     A,10
7FD7 B7         00590           JR      A
7FD8 C9         00591           RET
                00592 ;
7FD9 7E         00593 AGC:      LD      A,(HL)
7FDA 23         00594           INC     HL
7FDB 1805       00595           JR      AGE
                00596 ;
7FDD 7C         00597 HEX:      LD      A,H             ; Hexa ki konverzio
7FDE CDE27F     00598           CALL    AGE
7FE1 7D         00599           LD      A,L
7FE2 F5         00600 AGE:      PUSH    AF
7FE3 1F         00601           RRA
7FE4 1F         00602           RRA
7FE5 1F         00603           RRA
7FE6 1F         00604           RRA
7FE7 CDEB7F     00605           CALL    AGF
7FEA F1         00606           POP     AF
7FEB E60F       00607 AGF:      AND     0FH             ; 1 Hexa karakter
7FED C690       00608           ADD     A,90H
7FEF 27         00609           DAA
7FF0 CE40       00610           ADC     A,40H
7FF2 27         00611           DAA
7FF3 C33300     00612 AGG:      JP      33H
                00613 ;
7FF6 3E0D       00614 CRF:      LD      A,0DH
7FF8 18F9       00615           JR      AGG
                00616 ;
7FFA 3E1E       00617 AGJ:      LD      A,1EH
7FFC 18F5       00618           JR      AGG
                00619 ;
7FFE 0000       00620 RPCE:     DEFW    0               ; trace pointer
                00621 ;
7C18            00622           END     STRT
00000 TOTAL ERRORS
256*4  TEXT AREA BYTES LEFT
```

5. lista. A BASIC-program. A memóriában levő programból a betöltőprogram formátumában BASIC DATA sorokat állít elő. A V értéke (40-es sor) gépfüggő, az END utáni első szabad memóriahely címe.

```
10 DEFINT A-Z
20 INPUT "KEZDO CIM"; K: K= K-1
30 INPUT "VEGCIM   ";E
40 V=28621
50 POKE V,0: POKE V+1,0: V=V+2
60 S=:010: I=0: U=0
70 S=S+10: POKE V,S-INT(S/256)*256: POKEV+1,INT(S/256):
80 POKE V+2,136: POKE V+3,32: V=V+3
90 I=I+1: U=U+I: A=PEEK(K+U): LH=INT(A/16): LL=A-LH*16
100 LX=LH: GOSUB 200: LX=LL: GOSUB 200
110 IF K+U=E OR I=32 THEN 130
120 V=V+1: POKE V,44: GOTO 90
130 I=0: J=V: FOR Z=1 TO 3: POKE V+2,0: NEXT Z: V=V+2
140 J=J-1: IF PEEK(J)=PEEK(J-1)+PEEK(J-2)<>0 THEN 140
150 POKE J,INT(V/256): POKE J-1,V-INT(V/256)*256
160 V=V+2: IF K+U=E THEN 170 ELSE 70
170 POKE 16722,V-INT(V/256)*256: POKE 16723,INT(V/256)
180 POKE 16632,PEEK(16722):POKE 16634,PEEK(16723)
190 STOP
200 V=V+1: IF(LX)9) THEN L$=CHR$(LX+55) ELSE L$=CHR$(LX+48)
210 POKE V,ASC(L$): RETURN
220 END
230 .....
    .....
    .....
240 .....
    .....
    .....
250 .....
    .....
    .....
260 .....
    .....
    .....
270 .....
    .....
    .....
280 .....
    .....
    .....
290 .....
    .....
    .....
300 .....
    .....
    .....
310 .....
    .....
    .....
320 .....
    .....
    .....
330 .....
    .....
    .....
340 .....
    .....
    .....
350 .....
```


Építészeti, tervezési feladatai megoldásához használja a

SOFTinvest

SZOFTVERKERESKEDELMI ÉS FEJLESZTÉSI BETÉTI TÁRSULÁS

PROGRAMTERMÉKEIT!

ArchiCAD™

3 dimenziós építész tervező rendszer IBM PC/XT/AT, ill. Apple gépekre.
A Nyugat-Európába már 1984 óta exportált program által biztosított lehetőségek közül néhány:

- alaprajz interaktív tervezése képernyőn grafikus tablettel vagy "mouse"-zal;
- automatikus méretezés, terület- és térfogatszámítás;
- magasságadatok numerikus bevitele;
- homlokzati, metszeti és perspektív nézetek automatikus szerkesztése;
- kiviteli terv szintű rajzok készítése plotterre tetszőleges méretarányban;
- konszignációs listák és költségkalkuláció automatikus nyomtatása.

SIKTA statikai-szilárdsági programcsomag Commodore—64-es gépre.
Acél- és vasbeton tartók méretezésének ellenőrzése a hatályos (1986-ban életbe lépett) szabványok figyelembevételével.

Kérjen bővebb tájékoztatót!

Címünk:
1391 Budapest, Pf. 218
Telefon:
129-230

A MINŐSÉGÜGY KÖ

Mit vegyünk figyelembe a gépválas

Mint már szó volt róla, *szoktak* „a minőség" mértékére vonatkozóan ítéleteket alkotni a konkrét felhasználó és a konkrét felhasználási folyamat jellemzőinek figyelmen kívül hagyásával, de *lehet* ilyen ítéleteket alkotni ezek figyelembevételével is. Mivel a minőség — véleményünk szerint — *megfelelőség, illeszkedés*, ezért azoknak az ítéleteknek a konkrét helyzetekben való hasznosíthatósága, amelyek a konkrét helyzetektől függetlenek, nem lehet nagyobb azoknak az ítéleteknek a konkrét helyzetekben való hasznosíthatóságánál, amelyek a konkrét helyzet jellemzőit is — helyesen! — figyelembe veszik, és nem „a minőségről" beszélnek, hanem egy konkrét célra vonatkozó és konkrét helyzetben tapasztalható minőségről, megfelelőségről.

Sorozatunk elején azt ígértük, hogy számítástechnikai eszközök és szolgáltatások minőségével és a számítástechnikai eszközök minőségügyi (pl. minőségirányítási) alkalmazásaival foglalkozunk. A *rendelkezésünkre álló hely egyre szűkebbre szabása* miatt második célunknak még az alapos exponálásáig sem juthattunk el. Most, a minőségüggyel való foglalkozásunk itteni utolsó alkalmával az első témakör fontos részproblémájához, a gépválasztási döntéshez adunk segítséget. Terjedelmi korlátok miatt nem sorolhatjuk fel részletesen a figyelembe veendő jellemzőket, meg kell elégednünk azzal, hogy a kidolgozott részletes ismérvrendszerből néhány címszóból álló válogatást közöljünk; ezt az Olvasónak kell majd adottságainak, körülményeinek megfelelően kiegészítenie és kifejtenie, amikor saját problémájának megoldásával foglalkozik.

Kommunikációs eszközök, műveletek és folyamatok

□ Embertől emberhez, gépi berendezésen keresztül (E—E tipus); □ Embertől gépi berendezéshez (E—G tipus); □ Gépi berendezéstől emberhez (G—E tipus); □ Gépi berendezéstől gépi berendezéshez (G—G tipus) kommunikáció jellemzői.

E—G tipus. *Kézzel való vezérlés:* billentyűzet (leütés), érintésvezérelt kódadó, érintésvezérelt képernyő (megérintés), rajzolótábla, digitalizáló, egér, mozgatógömb, botkormány, fényceruza (mozgatás). *Lábbal való vezérlés:* pedál (lenyomás). *Hanggal való vezérlés:* beszéd.

G—E tipus. *Látható jeleket adók*, képernyők, különféle fényjeleket kibocsátó (nem ernyő jellegű) megjelenítők, papírra, műanyagra nyomtatók (hagyományos, tintasugaras stb.) rajzológépek. *Hallható jeleket adók:* zümmögők, zenei hangokat adók, beszédszintetizátorok.

G—G tipus. Lyukszalaglyukasztók, lyukszalagok, lyukszalagolvasók; lyukkártyalyukasztók, lyukkártyák, lyukkártyaolvasók; mágnesesszalag-írók, mágnesesszalagok, mágnesesszalag-olvasók (író és olvasó általában egybeépítve); hajlékony (mágneses)lemez-írók, hajlékony lemezek, hajlékony mágneseslemez-olvasók (az író és az olvasó egybeépítve); mágneseskártya, mágnesescsík-írók, . . . , széllyukasztott kártya lyukasztók stb. Elektromos kódadók, elektromos vezetékek, elektromos kódformálók és vevők, aktív memóriakártyák, rádiókódadók és -vevők, műholdas információadás, továbbítás és vétel eszközei, optikai leolvasók (pl. írásolvasók), fénykódadók, fénykábelek, fénykódvevők és feldolgozók.

Minden kommunikációra igaz, hogy minden bemeneti berendezés érzékelő, vevőeszköz, minden kimeneti pedig adó. (Természetesen nem foglalkozhattunk minden érzékelővel és minden adóval, ami a gyakorlatban szerepel.

Feldolgozási eszközök, műveletek, folyamatok

Hardver; műveleti sebességek, pontosságok.

Szoftver; operációs rendszerek, fordító és értelmező programok, különféle célra szolgáló programok és programrendszerek.

Hardver és szoftver bővítési lehetőségek.

Karbantartási, üzemeltetési, megbízhatósági tulajdonságok.

Tárolóeszközök

Lyukszalagok, lyukkártyák, mágnesesszalagok, (mágnesescsík, mágneseskártya stb.), hajlékony mágneseslemez, merev mágneseslemez, félvezetős tárak, tárpatronok, kapszulák (fix és változtatható tartalmúak).

Kiegészítő eszközök, berendezések, szolgáltatások

Hőmérsékletszabályozók, páratartalomszabályozók, légszűrők, egyéb zavarelhárítók.

Tárolóterületek pl. mágneses információhordozók számára, adatelőkészítés stb., számára speciális bútorok és berendezési tárgyak stb.

Gépkönyvek, dokumentációk, kiképző és továbbképző tanfolyamok (szóban, nyomtatásban, hajlékony lemezen stb.)

Javítószolgálat, felhasználói egyesületek, klubok, szaksajtó.

Üzemeltetési sajátságok, követelmények.

Szavatossági és más pénzügyi lehetőségek, illetve adottságok.

Referencia (hardver, szoftver stb. vonatkozásban).

A gép szűkebb (felhasználó) környezetének jellemzése, amelynek céljai eléréséről van szó

A feldolgozási és tárolási igények jellemzése pénzügyi, megbízhatósági stb. szempontból.

A felhasználó környezet jellemzése létszám, szakképzettség, érdeklődés, ellátottság, lélektani, szociális stb. helyzet és alakulás szempontjából.

A használati (üzemeltetési) módok jellemzése.

Az elérendő célok jellemzése.

*

E rövidített cimszótömörítmény után még a jó minőségű döntéselőkészítéshez a következő általános megfontolásokat is figyelembe kell vennünk.

Az alkalmazkodás figyelembevétele

A minőség, a megfelelőség mértéke — felfogásunk szerint — az igény és az igénykielégítő illeszkedésének valamilyen mértéke. Nagyon fontos azonban azt is figyelembe vennünk, hogy esetünkben még közelítőleg sem vehetjük merevnek sem az igényt (igénylőt, felhasználót), sem az igénykielégítőt. (Nem olyan a helyzet, mint pl. egy csap és egy furat esetében.) A számítástechnikai berendezés környezete (amelybe kerül) *alkalmazkodásra képes és alkalmazkodik is.* Sőt, a gépben magában is lehetnek tanuló—alkalmazkodó részrendszerek, például ilyen jellegű programok. Nem közömbös tehát az illeszkedés (időben nem szükségképp változatlan!) mértékének megállapításánál a felhasználó és a felhasznált (általában egymástól és más tényezőktől is függő) *alkalmazkodási folyamatok körültekintő, sokszempontú* (pl. pénzügyi, munkalélektani stb.) *minősítése és figyelembevétele.* Vizsgálnunk kell például az igénykielégítő számítástechnikai komplexum — hardver, szoftver stb. — megtanulhatóságát, használatának biztonsági, hibázási és ezzel rokon jellemzői mellett (és vele összefüggésben) a felhasználónak pl. tanulékonysági és munkafegyelmi jellemzőit is.

Többszempontú megfelelőség

Az eddigiekből nyilvánvaló, hogy mindig több különböző szempontból, de nem egymástól elkülönítve kell a megfelelőségekkel foglalkoznunk. Sőt, mivel a minősítendő rendszer maga is több egységből (részrendszerből) tevődik össze, így ilyen többkomponensű rendszerek működésének jóságát is értelmeznünk és mérnünk kell. *Rendkívül fontos, hogy elkerüljük a közkeletű tévedést, mely szerint a legjobb rendszert az elérhető legjobb komponensek beszerzésével építhetjük fel.* A komponensek közötti *harmónia* nagyon fontos minőségmeghatározó, de csak nagyon lazán és közvetetten függ a komponensek (önmagában mért!) minőségétől.

A harmonikusság fontosságát jól szemlélteti az az eset, amikor az igényelt feladatmegoldásokat például kb. 10 közepes professzionális személyi géppel lehet megoldani. Hogyan döntsünk ezek típusára vonatkozóan? (Nem kell homogénnek lennie a gépállománynak. Lehet benne több gyártó, több különböző típusa is.)

GY

snál?

Világos, hogy a harmónia függ a működtetési területtől, módtól és sok mástól is, és szinte minden felhasználási jellemzőre hat.

Van golyóscsapágy-gyártási technológia, amely nem nagyon törődik a golyók méretének „állandó értéken tartásával". A golyók és a hely, ahová kerülnek, nem lenne harmonikus, jó együttműködésre képes rendszer, ha a golyókat találomra választva szerelnék össze a csapágyakat. Nagymértékű harmóniát — és jó minőséget — érnek azonban el a golyók méretének és annak a helynek pontos mérésével, ahová a golyók kerülnek; a mérések alapján válogatnak, és minden golyó a neki legmegfelelőbb helyre kerül.

Hasonló — de bonyolultabb — a legnagyobb mértékű számítástechnikai harmónia kialakításának módszere is.

Az idő szerepe

Nem könnyű valaminek valamilyen célra való pillanatnyi megfelelőségét sem mérni, megítélni. Nehezebbé válik azonban a mérési, megítélési feladat, ha a szóban forgó valami tulajdonságai nem állandók, hanem változnak az időben, és a megfelelőségi ítélet kialakításánál nem egy időpontot, hanem több időpontból álló összességet, például időszakot, időtartományokat kell figyelembe vennünk. Ez a helyzet például a gépvásárlásnál is: az üzembe állítástól a leszerelésig (kiselejtezésig) tartó időszak egészét figyelembe véve kell „globális" minőségi ítéleteket alkotnunk, és ezek az ítéletek nem azonosak a pillanatonkénti megfelelőségeknek a szóbanforgó időszakbeli összességével. Még nehezebb lesz a problémánk, ha az előbbiek mellett még a cél és a felhasználás módja is változik a vizsgált időszakban. A minőségügyben (máshol nem mindig!) a legnehezebb a legáltalánosabb és a legvalóságosabb probléma: az, amikor minden változik az alapul vett időtartományban, méghozzá nem is egyástól függetlenül. A felsoroltakon kívül változik még például az a mód, illetve algoritmus is, amellyel a megfelelőséget bíráljuk, a minőség mértékét megállapítjuk, összefüggésben más, a minőségi ítéletet befolyásoló folyamatokkal.

*

Eredeti célkitűzésünkben fontos szerepet szántunk a számítástechnikai megnyilatkozások minőségével való foglalkozásnak is, így például a vadhajtások nyesegetésének, és a torzulásokra való figyelemfelhívásoknak is. Ezt a célt szolgálta a „Zöldségeskert" rovat.

Remélve, hogy a minőségügy előbbrevitelének útján nemcsak rögök és kátyuk, hanem jó minőségű útszakaszok is lesznek majd, kívánja az Olvasónak az itt megkezdett munka sikeres folytatását „A minőségügy közügy" és a „Zöldségeskert" rovatot javasló és író

POGÁNY CSABA

MIKROSZÁMÍTÓGÉP-GYÁRTÓK, -FELHASZNÁLÓK FIGYELMÉBE

$ helyett Ft!

1. ajánlatunk:

WINCHESTER DISC CONTROLLER
SCSI (SASI) interface
– max. 4 db tetszőleges típusú és kapacitású
– 5,25" Winchester disc drive vezérlése
– funkcionálisan kompatibilis a XEBEC S 1410 ill.
WD 1002–SHD vezérlőkkel

Ár: tartozékokkal 59 900 Ft

2. ajánlatunk:

IBM KOMPATIBILIS FLOPPY DISC DRIVE CONTROLLER

Ár: tartozékokkal 18 900 Ft

Szállítás raktárról
12 hónap garancia

ÁRENGEDMÉNY
nagyobb darabszámú rendelés esetén

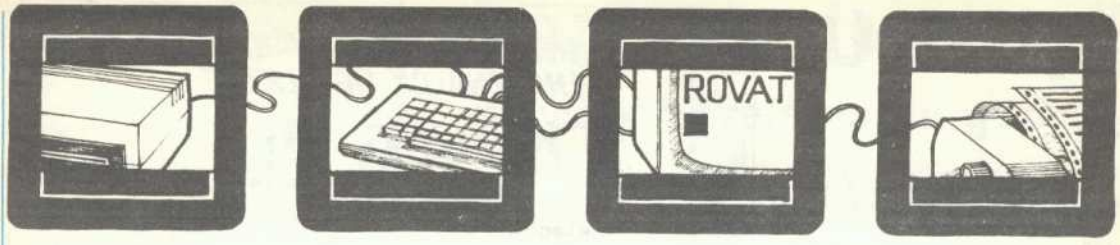
Felvilágosítás:
Gigor Györgyné
Telefon:
693-253
Levélcím:
1369 Budapest Pf. 317

mikromatika

Nemzetközi Számítástechnika
– Internacia Komputado

Az 50 országban olvasott magazin példányonként 30 forintért kapható az SKV Könyvesboltban

Bp. II., Keleti Károly u. 10. Telefon: 158-018



DIÁK C—16 ROVAT

Öninduló program

A $0770-en kezdődő program először átírja a BASIC MELEGSTART vektort, majd szalagra menti a $02A7—$03Ø3 területet. A mentést azért nem a monitor S parancsával végzem, mert csak így kerül a helyére BASIC-ből történő beolvasás alkalmával is.

Ha ezután beolvassuk ezt a programot, akkor az átírja a fent említett vektort, és ahogy az olvasással elkészül a gép, a vezérlést a $02A7-re adja. Ez először visszaírja a vektor eredeti értékét ($8712), majd sorra hívja a SETLFS, SETNAME és LOAD KERNAL rutinokat, ami megfelel a BASIC LOAD funkciójának. Ezután a $8A93-at hívja, ami CLR-nek felel meg, és kötelező meghívni az őt követő $8BCD előtt, amely a RUN-nak felel meg. Ez utóbbit nem JSR-rel, hanem JMP-vel hívom, így egy RTS elhagyható.

Tapasztalatom szerint ezt a programot nem célszerű a $02A7-nél feljebb tenni, mert előfordulhat, hogy elszáll.

Írjuk be monitor üzemmódban az *1.* és *2. listán* látható két gépi rutint, majd adjunk ki egy SYS1Ø4 parancsot. Ezt követően a gép a SAVE utasításnál megszokottak szerint jár el. Körülbelül 8 fordulatnyi helyet vesz igénybe a szalagon, és az ezt közvetlenül követő programot teszi önindulóvá oly módon, hogy a beolvasássorban a LOAD után nem áll meg, hanem tovább olvas, majd a beolvasás után végrehajt egy RUN parancsot. Így ha a következő program RUN-nal indítható, akkor az azonnal elindul. Ha a BASIC-program POKE8Ø6,1Ø3 utasítással kezdődik, akkor az a továbbiakban sem állítható le.

CSELÉNYI ZOLTÁN

1. lista

```
; FF00 00 00 FF 00 F8

. 02A7  A9 12     LDA #$12
. 02A9  8D 02 03  STA $0302
. 02AC  A9 87     LDA #$87
. 02AE  8D 03 03  STA $0303
. 02B1  A9 01     LDA #$01
. 02B3  AA        TAX
. 02B4  A0 FF     LDY #$FF
. 02B6  20 BA FF  JSR $FFBA
. 02B9  A9 00     LDA #$00
. 02BB  20 BD FF  JSR $FFBD
. 02BE  A9 00     LDA #$00
. 02C0  20 D5 FF  JSR $FFD5
. 02C3  20 93 8A  JSR $8A93
. 02C6  4C CD 8B  JMP $8BCD
```

2. lista

```
. 0770  A9 02     LDA #$02
. 0772  8D 03 03  STA $0303
. 0775  A9 A7     LDA #$A7
. 0777  8D 02 03  STA $0302
. 077A  A9 01     LDA #$01
. 077C  AA        TAX
. 077D  A0 FF     LDY #$FF
. 077F  20 BA FF  JSR $FFBA
. 0782  A9 00     LDA #$00
. 0784  20 BD FF  JSR $FFBD
. 0787  A9 02     LDA #$02
. 0789  85 D3     STA $D3
. 078B  A9 A7     LDA #$A7
. 078D  85 D2     STA $D2
. 078F  A9 D2     LDA #$D2
. 0791  A2 04     LDX #$04
. 0793  A0 03     LDY #$03
. 0795  20 D8 FF  JSR $FFD8
. 0798  60        RTS
```

PRIMO — Átszámozó program

Aki egy számítógépen megismerte a RENUMBER, RENUM, RE parancsot, más gépeken hiányolja. Egy komolyabb program fejlesztése közben az átírások, betoldások, átrendezések után eléggé rendetlen sorszámozás alakul ki: távol áll az eszményi 10-es növekménytől!

A következő kis program átszámozást végez. Azt természetesen nem várhatjuk egy ilyen egyszerű segédeszköztől, hogy átírja a GOTO-k és GOSUB-ok sorszámát is: ezt nekünk kell utólag elvégezni. De még így is érdemes lehet alkalmazni hosszú programok átszámozására, főleg ha nincs bennük túlságosan sok vezérlésátadás.

Először billentyűzzük be nagyon figyelmesen az *1. listát*, majd futtatás, kipróbálás nélkül vegyük fel kazettára. Listázással ellen-

1. lista

```
60000 REM PRIMO ATSZAMOZO
60010 DEFINT A-H: A=234: B=67: D=255
60020 E=255: POKE 16548,A,B
60030 INPUT "KEZDOSZAM";G
60040 INPUT "NOVEKMENY";H
60050 C=A+B*E: A=PEEK(C): B=PEEK(C+1)
60060 F=PEEK(C+2)+PEEK(C+3)*E
60070 IF F=60000 THEN 60100
60080 POKE C+2,G AND D,G/E
60090 G=G+H: GOTO 60050
60100 BEEP 500,100: PRINT "KESZEN VAN."
60110 DELETE 60000-60110
```

A Spectrum rendszerváltozói

DIÁK spectrum ROVAT

Az előző részben a kényelmi szempontokat kiszolgáló változókról volt szó. Ezt a kört kibővítem a programozáshoz használható rendszerváltozókkal. A programba beépítve sok olyan funkció kialakítható, amely a gép BASIC nyelvében nem érhető el.

HEX	DEC	NÉV	FUNKCIÓ
5C3A	23610	ERR NR	A hiba kódját tartalmazza. Kezdeti értéke 255, ekkor a Ø OK felirattal adja vissza a kurzort a gép a programozónak. Látható, hogy a hibajel kódját csökkenteni kell eggyel, ez adja a változó értékét. Egyszerű üzenetközvetítést tesz lehetővé. Ha a program utolsó sorában beírunk ebbe a bájtba egy kódot, akkor hibaüzenettel áll le a program.
5C48	23624	BORDCR	A keret (BORDER) és az alsó szerkesztő sorok attributumait (színezéseit) tartalmazza. Az alsó 3 bit a szerkesztő sorok tintaszíneit, a következő 3 bit a keret és az alsó sorok papírszínét adja. Átírásával ezeket a tintaszíneket érdemes módosítani, mivel ezt nem végzi el a BORDER utasítás, nem teszi lehetővé a tetszőleges színválasztást.
5C6B	23659	DF SZ	A szerkesztő sorok számát tartalmazza. Értéke alapállapotban 2. Ø-t bepoke-olva az interpreter nem talál szerkesztő sort, ezért nem tudja visszaadni a kurzort. Ezt a tulajdonságot programok titkosítására lehet felhasználni.
5C6E	23662	OLDPPC	A következő végrehajtandó programsor sorszáma.
5C7Ø	23664	OSPPC	A sorban a következő végrehajtandó parancs száma. A két változó átírásával kiszámított GOTO utasítást hajthatunk végre, akár egy sor közepére is.
5C76	23670	SEED	Az RND függvény értéke. A RANDOMIZE utasítás ezt írja át. Ha van operandusa, akkor azt írja be, ha nincs, akkor a Spectrum belső óráját másolja át.
5C78	23672	FRAMES	A Spectrum belső órája. Minden 20 ms-ban előre számol egyet. Ha a megszakítás tiltva van, akkor értéke állandó, nem számol. Ez általában a magnókezeléskor fordul elő.
5C7B	23675	UDG	A saját definiálású grafikus karakterek generátorának kezdőcíme. Értéke megkapható a PRINT USR "A" paranccsal. A könnyebb kezelhetőség érdekében az UDG karaktereket érdemes az első REM sorba helyezni, a 23760-as címtől beadogatni, és erre átírni ezt a rendszerváltozót.

Megjegyzés: az első részben a REPDEL változó címe 23561.

HEX	DEC	NÉV	FUNKCIÓ
5C7D	23677	COORDS	Az utolsó megrajzolt pont koordinátája. Különleges grafikai rutinok (boxrajzolás, FILL funkció stb.) megírásánál adhat segítséget, mivel a BASIC itt adja át és veszi át a grafikai koordinátát. Az első bájt az X, a második az Y koordináta.
5C84	23684	DF CC	A következő PRINT pozíció kezdőcíme a képernyő-memóriában. Gépi kódú sprite-kezelés esetén használható fel, ha az (X,Y) koordinátákba (a következő részben) betöltjük a megfelelő adatokat; ekkor itt kapjuk meg a kezdőcímet.
5C8C	23692	SCR CT	A scrollozható sorok számát tartalmazza. Ha értéke Ø lesz, kiírja a SCROLL? feliratot, majd gombnyomásra feltolja a sort, és rendszerváltozóba 22-t ír be. Folyamatos scrollozáshoz értékét állandóan 1 felett kell tartani.
5C8D	23693	ATTR P	Az állandó színeket tartalmazza. Az alsó 3 bit a tintaszínt, a következő 3 bit a papírszínt adja. Program közben ennek kiolvasásával lehet meggyőződni az aktuális színekről.
5C8F	23695	ATTR T	Az ideiglenes, egy PRINT, PLOT stb. utasítás idejére aktuális színeket tartalmazza, hasonlóan az ATTR P változóhoz.
5CBØ	23728	NMI ADD	A ROM-ban elrontott nem maszkolható megszakítás rutin ugrási címe. A Bit-Letben közölt megszakító áramkör alkalmazása esetén ide kell beírni az ugrási címet.
5CB2	23730	RAMTOP	A BASIC által elérhető memóriaterület felső bájtjának címe. A CLEAR utasítással lehet felülírni.
5CB4	23732	P—RAMT	A létező RAM-terület utolsó bájtjának címe. Bekapcsoláskor a Spectrum ellenőrzi memóriájának végét. Memóriahiba esetén a 23733-as címen nem 127 (16 k) vagy 255 (48 k) található, hanem kisebb szám.

RUSZNYÁK GÁBOR

```
A=PEEK(16633)+PEEK(16634)*256-2:
POKE 16548,A AND 255,A/256
```

2. lista

őrizhetjük kimentés előtt, de indítással nem. A felhasználás menetrendje:

1. Töltsük be a gépbe az átszámozandó programot!
2. Írjuk be a *2. listán* látható parancsokat!
3. Töltsük be az átszámozó segédprogramot (a 2. lista parancsainak hatására az a főprogram mögé kerül, ezért kellett magas sorszámokat alkalmazni).
4. Indítsuk el RUN-nal! Először kéri a kezdőértéket és a számozás lépésközét: mindkettőt igényeink szerint adhatjuk meg.

Hang és kiírás jelzi, ha készen van az átszámozás. Utána ne feledkezzünk meg az ugróutasítások átírásáról sem!

FEKETE GYÖRGY

SOFTCOPY CBM MPS802-RE

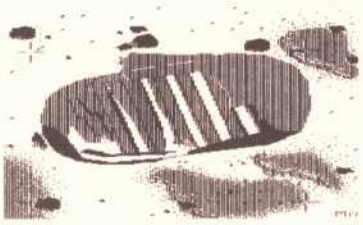
```
1 :
2 REM ************************
3 REM **                      **
4 REM **  S O F T C O P Y     **
5 REM **                      **
6 REM **    SCHADT GYORGY     **
7 REM **       (C)1986        **
8 REM **                      **
9 REM ************************
10 :
50 BLOKK=
52 GOSUB1000
54 END
1000 :
1002 REM ************************
1004 REM **                      **
1006 REM ** SOFTCOPY SZUBRUTIN  **
1008 REM **                      **
1010 REM ************************
1012 :
1014 OPEN6,4,6:PRINT#6,CHR$(20):CLOSE6
1016 OPEN5,4,5:OPEN4,4
1018 SC=1024*BLOKK
1020 FORI=0TO39
1022 FORJ=24TO0STEP-1
1024 FORK=7TO0STEP-1
1026 C=SC+I*8+J*40*8+K
1028 PRINT#5,CHR$(PEEK(C));:NEXTK
1030 PRINT#5:PRINT#4,TAB(24-J)CHR$(254)CHR$(141);:NEXTJ
1032 PRINT#4:NEXTI
1034 CLOSE4:CLOSE5
1036 OPEN6,4,6:PRINT#6,CHR$(24):CLOSE6
1038 RETURN
1040 :
READY.
```

Régi problémám, hogy képtelen vagyok nagyfelbontású grafikák nyomtatására az MPS802-vel. A SIMON'S BASIC, a SUPERGRAFIK és még egy csomó program csak karakterek zagyva összevisszaságát produkálja. Eddig még nem találkoztam olyan programmal, mely képes grafikák kivitelére erre a típusú nyomtatóra, bár hirdetésekben olvastam, hogy léteznek ilyenek. A MPS802 ugyanis nem a C64 karaktereit használja. Megtalálható rajta a teljes grafikus karakterarzenál is, de az írásjelek új — nyomtatón szebben mutató — formát kaptak. A gép tervezői a felhasználónak csak egyetlen programozható karaktert hagytak (CHR$(254)).

A program működése

Tudjuk, hogy a nagy felbontású képernyő 8 k-t foglal el. A BLOKK változó a szubrutin egyetlen bemenő adata, tartalma azt mutatja meg, hogy hányadik k-nál kezdődik ez a 8 kbájt. Beállítása az 50-es sorban.

1014	A sorok közti távolság beállítása 0-ra.
1018	A startcím kiszámítása (SC változó).
1020—1024	Ciklusok: I=oszlopszám, J=sorszám, K=bájtok egy karakteren belül.
1026	Aktuális cím kiszámítása (C változó).
1028	A programozható karakter adatainak kivitele.
1030	PRINT 5 után a karakter új formája lesz érvényes. A karakter kiíratása, CHR$(141) jelentése: kocsi vissza soremelés nélkül. Azért kell, mert egyébként soremelés történne.
1032	Soremelés.
1036	Sorok közti távolság visszaállítása szöveg nyomtatásához.



Lépésnyom a holdon

esetében már a kezdőknek is megy, de például a SIMON'S BASIC-nél erre még nem tudtam rájönni.

Végül egy jó tanács: sötét háttér és világos pontszín esetén az 1028-as sorba írjunk CHR$(155-PEEK(C));-t (inverz nyomtatás).

SCHADT GYÖRGY

Lányok

Az ábrák a PAINT MAGIC demorajzai. A rajtuk látható keresztirányú és függőleges sötét csíkok a papírtovábbítás és írófejmozgatás pontatlanságából adódnak. Egyébként a képek multicolor felbontással készültek, mégis jól mutatnak. Ha egy képet ki akarunk nyomtatni, tudnunk kell, hogy hol kezdődik. Ez saját programjaik

Naplemente

FORTH rekordszerkezetek II.

Az előző számunkban félbehagyott cikkünkhöz kapcsolódik a következő feladat.

Hogyan lehetne a TELJES és a RECIM szavakat úgy megírni, hogy csak REK, illetve REK2 típusú szavakat fogadjanak el? Írjunk mondjuk egy REK? nevű szót, veremhatása: (paramzőcím ———), amely megszakítja a programfutást, ha a vermen talált cím a REK-kel definiált egyik rekord paramzőcíme.

Oldjuk meg most, hogy az egyes mezőkre ne sorszámmal, hanem névvel lehessen hivatkozni. A mezőnév a mező címét tegye a verembe (tehát ő és ne a rekord legyen a mezőcímszó, a rekordból pedig valahogy össze lehessen szedni a mezőneveket. Továbbá, ha a rekord kezdőcímét meg akarjuk változtatni, azt egyetlen helyen kelljen, s a mezők azontúl mégis mind az új kezdőcímet használják. Ez az egyetlen hely a rekord paramzője lesz, innen veszik majd a mezők az éppen aktuális kezdőcímet.

A mezők paramzőjében a relatív cím lesz, meg egy pointer a rekord címére (a rekord paramzőjébe), vagy másképpen a rekordcím címe.

A mezőket definiáló szó, a MEZŐ a relatív címeket ugyanúgy a mezőhosszokból fogja számítani, mint az eddigi rekordok. Az első relatív címet mi fogjuk megmondani (az úgyis 0), azontúl pedig minden MEZO definíció a következő mezőhossz és az előző relatív cím összegét, azaz az új relatív címet adja tovább a vermen. A MEZO veremhatása:

(mezőhossz relcím ——— újrelcím).

A mezőket a hozzájuk tartozó rekord előtt definiáljuk, amikor a rekord paramzőjébe mutató rekordcím címének értékét még nem tudhatjuk. Így a rekordcím címét utólag fogjuk beírni, meződefiniáláskor csak helyet hagyunk neki.

```
: MEZO
    <BUILDS    ( mezohossz relcim --- ujrelcim )
       DUP  ,      ( letesszuk a relativ cimet )
       +           ( a masik peldanyt noveljuk az eppen )
                   (     definialt mezo hosszaval )
       2 ALLOT     ( hely a rekordcimcim szamara )
     DOES>  ( ---- mezocim )
       DUP @       ( a paramzo elejerol kiolvassuk a rel. cimet)
       SWAP 2+ @   ( rekordcimcim )
       @           ( rekordcim )
       +           ( rekordcim + rel.cim  -- keszen vagyunk)
```

A rekordot definiáló szó, a REKORD a MEZŐ hívogatásával létrehozza a mezőket, majd a megszokott <BUILDS, DOES> szópárral a rekordot. A rekord létrehozása után beírja a rekordcímet a mezők paramzőjébe. A REKORD veremhatása:

(mezőhossz$_{msz}$ ... mezőhossz$_1$ msz rekordcím ———).

A mezők és a rekord nevét így adjuk meg:

REKORD mezőnév$_1$ mezőnév$_2$ ... mezőnév$_{msz}$ rekordnév (abban a sorrendben, ahogy ezek a szavak létrejönnek).

A rekord paramzője (zárójelben a paramző elejéhez képest relatív cím):

2 bájton a rekord kezdőcíme (0)
1 bájton a mezők száma (2)
2 bájton a rekordhossz (3)
2 bájton az utoljára definiált mező névmezőcíme (ezzel lehet majd kezdeni a mezők felgöngyölítését) (5).

Így nézne ki egy rekord a könyvkölcsönzőnél: (mondjuk, a PAD-et használja tárnak):

6 80 60 (mezőhosszok) 3 (mezők száma) PAD (rekordcím)
REKORD CIM KINEK DATUM KOLCS3

Itt az egyes mezők címét a CIM, KINEK, DATUM szavak adják. A KOLCS3, a rekord pedig nem tesz mást, mint hogy a vermen hagyja paramzőjének a címét. Így nem kell a —FIND-del trükköznünk, ha soron kívüli szolgáltatásokat akarunk írni a rekordunkhoz.

Utolsó rekordunk előtt írunk egy segédszót, amely egy szótári szó névmezőcíméből az előtte definiált szó névmezőcímét adja:

```
: VISZ    ( nfa ---- elozo-nfa )
     PFA  ( a lancmezo cimet csak a paramzobol kaphatjuk meg )
     LFA @  ( a lancmezo az elozo nevmezocimet tartalmazza )
```

```
: REKORD    ( hossz-msz  ... hossz-1 msz rekordcim --- )
      >R >R     ( keszenletben a mezohosszok )
      0         ( az elso mezo relativ cime )
      R 0 DO  MEZO LOOP  ( letrehoztuk a mezoket )
      R> R>     ( rendet csinalunk a return stacken )
        ( a vermen: rekordhossz msz rekordcim )
    <BUILDS     ( most szuletik a "rekord")
      HERE      ( ide kerul a rekordcim,tehat ez a rekordcimcim,)
                (    amelyet majd beirunk a "mezok" paramzojebe )
      SWAP  ,   ( beirjuk a rekordcimet ) -->


      OVER C,   ( ,meg a mezoszamot )
      ROT   ,   ( ,meg a rekordhosszot. A vermen: msz rekordcim )
      LATEST    ( az utoljara definialt szo - azaz a "rekord" )
                (       nevmezocime )
      VISZ      ( vissza az utolso "mezore" )
      DUP ,     ( nfa a paramzobe )
        ( most beirjuk a rekordcimcimet a "mezok" paramzojebe)
        ( a vermen: msz rekordcimcim nevmezocim)
      ROT 0     ( annyiszor ismetlunk, ahany mezo van )
      DO        ( a vermen: rekordcim nevmezocim )
        OVER OVER PFA
        4 +     ( a paramzobol 2 byte a FORTH sajat celjaira, )
                ( 2 pedig a rekordcim )
        ,
        VISZ    ( kerjuk az elozo mezot )

      LOOP      ( mar csak a felesleget kell letakaritani a )
      DROP DROP       ( vereemrol )
    DOES>       ( itt semmi feladat nincsen )
```

Ebben a rekordban könnyű az adatterület címét átállítani, hiszen a rekord neve éppen azt a címet adja a vermen, ahol az új rekordcímet tárolnunk kell.

Igaz-e, hogy ebből a rekordból össze lehet szedni, hogy milyen mezőkből áll? A rekord paramzője tartalmazza az utolsó mező névmezőcímét, meg azt is, hány mező van — mivel tudjuk, hogy a mezők folyamatosan, egymáshoz láncolva kerültek a szótárba, több már nem kell nekünk. Meggyőződésünket látványosabban is igazoljuk: írunk egy szót, amely kilistázza a rekord mezőit. A vermen természetesen a rekord paramzőcímét adjuk meg (a rekord nevével). A kiíró szó neve .REK.

Tehát például
KOLCS3 .REK
eredményeképpen a következők fognak a képernyőn megjelenni:
rekordcím: 5FD5
mezők:

```
rekordcim :5EF4
mezok:
     DATUM          relativ cim: 140   hossz :6
     KINEK          relativ cim: 60    hossz :80
     CIM            relativ cim: 0     hossz :60
```

DÁTUM relatív cím: 140 hossz: 6
KINEK relatív cím: 60 hossz: 80
CIM relatív cím: 0 hossz: 60

ha a rekordcím értéke történetesen hexadecimális 5EF4.

A mezőnév és az utána írt szóközök együtt mindig 16 karaktert tesznek ki, ettől kerülnek a "relatív cím" kiírások szépen egymás alá. Hogy ezt elérhessük, ismernünk kell az OUT változónak azt a tulajdonságát, hogy minden EMIT-nél eggyel növekszik. A képernyőre író FORTH alapszavak pedig valamennyien az EMIT-ot hívogatják (így az ID . . ." SPACES is, amelyeket használni fogunk). Ha az OUT változóba például a mezőnév kiírása előtt 0-t teszünk, akkor a kiírás után tudhatjuk, hány szóközzel lehet a kiírt karakterek számát 16-ra növelni. Mivel ilyen formára igazításra kétszer is szükségünk lesz, írunk rá egy KIEG nevű szót. A KIEG-nek megadjuk a vermen, hogy hányra kell növelnie a kiírt karakterek számát, ő pedig kiírja a szükséges számú szóközt.

```
--
: KIES  ( n ---- )
     OUT @ -   ( a kivant karakterszambol kivonjuk a meglevot )
     0 MAX     ( vigyazunk, hogy ne legyen hotrany akkor sem, )
               (          ha a meglevo karakterekkel mar tulleptuk )
               (          a keretet )
     SPACES
;

: .REK   ( paramzocim ---- )     CR  ." rekordcim :"
     BASE @   ( hexadecimalisan iratjuk ki a rekord cimet,)
              (   utana majd vissza akarjuk allitani a BASE -t )
     HEX      ( most rontottuk el BASE -t )
     OVER @   ( a paramzocim a rekordcim cime )
     U. BASE !     ( kiirjuk a cimet, visszaallitjuk BASE -t)
     CR ." mezok:"
     DUP 5 + @     ( utso mezo nevmezocime )
     OVER 3 + @    ( rekordhossz; kell a mezohosszok)
     PAD !         (              szamitasahoz )
     OVER 2 + C@   ( mezok szama )
     0 DO          ( a vermen: paramzocim nevmezocim )
          CR 5 SPACES
          0 OUT !   DUP ID. 16 KIEG ( mezonev kiirasa 16 hosszon)
          ." relativ cim: "
          DUP PFA   ( a rel. cimet a "mezo" paramzojebol )
                    (     vesszuk )
            2+      ( kikeruljuk a FORTH sajat celjaira szolgalo)
                    (     ezot )
          @         ( megvan a relativ cim )
          DUP .
          35 KIES  ." hossz :" '
          PAD @     ( a pad -ben a kovetkezo mezo relativ cime)
          SWAP DUP PAD !   ( gondoskodunk rola, hogy a kovetkezo)
                           (   kornel is igy legyen )
          - .       ( a ket rel. cim kulonbsege a hossz )
          VISZ      ( visszalepunk egy mezovel )
     LOOP
     DROP
```

Végül következzék egy újabb feladat.

Ha a rekord például egy AREA szóval definiált területen van, akkor a hexadecimálisan kiírt rekordcím kevesebbet mond nekünk, mint a terület neve. Jó lenne, ha a .REK ezt a nevet is kiírná. Írjunk egy .NEV? szót, amely megvizsgálja, hogy a vermen talált cím egy szótári szó paramzőcíme-e. Ha igen, akkor kiírja ennek a szónak a nevét. A .NEV? veremhatása: (cím ———).

Akinek van megoldása valamelyik feladatra vagy mindkettőre, küldje be a szerkesztőség címére. A boritékra írja rá: Programozástechnika rovat. Aki megírja a saját címét is, annak válaszolunk. A legszebb megoldásokat közöljük.

ADORJÁN NOÉMI

Z80 pro

A Spectrum-tulajdonosok többsége valószínűleg már találkozott a Hisoft-cég assemblerével és monitorával, a GENS3-mal és MONS3-mal. Ennek a két programnak nincs töltési és indítási címe: a tárba bárhova betöltve és elindítva működnek.

Hogyan lehet ilyen programot készíteni? Ha egy programot az n1 címre fordittatunk, de az n2 címre töltve akarjuk használni, akkor minden abszolút címre hivatkozás értékéhez hozzá kell adni n2—n1 értéket. Ennek legegyszerűbb esete az, ha n1=0, ekkor n2—n1=n2 a betöltési cím. Ha tehát egy assembly programot nulla kezdőcímmel fordittatunk, azaz a program elején az ORG direktíva argumentuma nulla, akkor minden, a programban szereplő cím értéke relatív a program kezdetéhez viszonyítva, és ezek az értékek kerülnek a tárgykódba; fordítás során minden, az ilyen címekre hivatkozó utasítás megfelelő két bájtjára.

Ha ezután a programot nem a nullás címre töltjük, akkor mielőtt elindítjuk, minden ilyen két bájthoz hozzá kell adni a betöltési cím értékét, hogy a betöltésnek megfelelő abszolút címre mutasson. Ezt a program elején elhelyezett ún. relokáló program végzi el. Valahonnan tudnia kell, hogy ez a két bájt hol helyezkedik el a programban. Címeiket például a program végén külön e célra elhelyezett relokációs táblázatból veheti. Természetesen ezek az értékek is a program elejéhez viszonyítva, relatívan szerepelnek. A relokáló program ráállít valamilyen mutatót erre a táblázatra, mindig kivesz egy értéket, ez a program elejéhez képest relatív, hozzáadja a betöltési címet, ekkor a címre hivatkozó utasítás vagy adat megfelelő szavának abszolút címét kapja. Erről a címről előveszi az ott lévő két bájtot, amelyek értéke relatív a program elejéhez viszonyítva, hozzáadja a betöltési címet, és visszateszi a most már abszolút címet. Ezután veszi a táblázat következő elemét, és ugyanezt végrehajtja mindaddig, míg a táblázat végére nem ér.

A relokáló rész így már félig megvan, csak a regiszterek megválasztása hiányzik. Két olyan regiszterpárra van szükség, amelyek arra a címre mutatnak, ahonnan, illetve ahova adatot kell kivenni vagy letenni. Az egyik a táblázatba mutató regiszterpár, a másik az, amelyben előállítjuk a megváltoztatandó szó címét. Az egyik lehet a HL, a másik a DE. A HL által mutatott címről bármely regiszterbe lehet értéket tölteni, és a HL-hez hozzá lehet adni például a BC-t, amelyben tárolhatjuk a betöltési címet. A DE könnyen megcserélhető HL-lel, így ezek a műveletek DE-vel is egyszerűen elvégezhetők.A Hisoft cég programozói is így választották meg a regiszterek szerepét. Jó gyakorlat a relokáló programot ilyen regisztermegválasztással megírni, vagy ha ez nem megy, akkor megérteni a korábban említett szoftverek működését disassembly lista alapján (lásd az *1. programot*).

Mi most egy másik megoldást nézünk meg. A táblázatból mindig kiveszünk két bájtot, és a mutatót növeljük kettővel, legközelebb pedig innen veszünk ki szót, és így tovább. Ha a táblázat elejére ráállítjuk a veremmutatót, akkor ez a művelet egy POP utasítás. HL lehet az a mutató, amelynek relatív értékét a táblázatból vesszük ki, DE-be tölthetjük a programból kivett két bájtot, és BC-ben lehet a betöltési cím. Így nincs szükség a regiszterek verembe mentésére sem.

Rendkívül fontos — és nem csak Z80 assemblynél —, hogy a relokáló programrésznek nem szabad egyetlen abszolút címre való hivatkozást sem tartalmaznia, vagy azt az első végrehajtás előtt relokálni kell.

A relokálás másik problémája, hogy meg kell tudnia a programnak a saját kezdő címét. A Spectrum gépen futó GENS3 és MONS3 ezt az értéket megkapja a BC-regiszterpárban. A Spectrumban az USR utasítás végrehajtása ugyanis a ROM-ban PUSH BC; RET utasításpárral fejeződik be. Ennél általánosabb módszer az, hogy meghívunk egy fix helyen lévő szubrutint, amely egyetlen

amozási gyakorlatok 5.

RET utasításból áll. Ilyet találunk bármely gép ROM programjában. A visszatérés után a verem alatt van a visszatérési cím két bájton. Két DEC SP utasítással a vermet „ráhúzzuk" erre a két bájtra, és így hozzáférhetővé válnak egy POP utasítással.

Ez így persze rendszerfüggő, hiszen ismerni kell egy RET utasítás címét. Ez akkor válik problémává, ha például egy CP/M rendszer alatt futó, a konkrét gép sajátosságait ki nem használó programot akarunk írni, azaz a rendszer szolgáltatásai közül csak a CP/M által deklaráltakat használjuk. Nem lehet olyan címet mondani, amelyen minden CP/M rendszerben C9h található. Igaz, meg lehet hívni a CP/M rendszert magát, de elképzelhető, hogy a CP/M kiveszi a veremből az értéket, JP (HL) utasítással tér vissza, és a verem alatt már valami egész más van. A *2. program* egy rendszerfüggetlen megoldást mutat.

A program, amit relokálunk, rendkívül egyszerű, a gépi kódú programok belövéséhez segítséget nyújtó program. Az első meghíváskor relokálja magát. Minden további, a programot meghívó CALL utasítás után el kell helyezni egy bájtot, például a DEFB direktíva segítségével. A program ekkor kiírja a regiszterek értékeit, valamint a CALL utasítás után elhelyezett bájt értékét, majd vár, amíg az A vagy C billentyűt le nem nyomjuk. A C gomb lenyomására visszakerül a vezérlés a CALL utasítás utáni bájt utánra, az A-ra a program futása megszakad.

Nézzük meg a relokáló részt!

A betöltési cím meghatározása

A program az ENTER címkével jelölt sortól indul. BC-be azért kerül nulla, hogy a CPIR utasítás az egész memóriát végignézze. Az akkumulátorba kerülő érték, amit a CPIR keres, annak az utasításnak a kódja, amely az SLFMCD címkéjű sorban van. A HL értéke mindegy, mert az egész memóriát végignézzük. A CPIR utasítás csak úgy állhat le, hogy talált megfelelő bájtot: ekkor HL a megtalált bájt utánra mutat. Ezt is összehasonlítjuk az akkumulátorral, hiszen az SLFMCD sorban szereplő utasítás két bájtját keressük, és mind a kettő 36h. Ha nem egyezik, akkor keresünk tovább. Ha találtunk két egymás utáni ilyen bájtot, akkor vagy a keresett két bájtot találtuk meg, vagy valamely, a memóriában szereplő másik két egymás utáni 36h bájtot.

Ennek eldöntésére a két bájtra elhelyezünk egy relatív ugróutasítást, és ugrunk a REMAKE címre. Ha valóban ezt a két bájtot találtuk meg, akkor most az SLFMCD sor utasítása helyén egy relatív ugróutasítás van a START címre. HL ekkor a relatív ugróutasításra mutat, ami néggyel több, mint a program kezdete. Ha nem erre a két bájtra akadtunk rá, akkor az SLFMCD sor az eredeti maradt, és ekkor az átírt két bájtot visszaírja a program az eredetire. A START címkénél HL-t négyszer csökkentjük, és áttöltjük BC-be. Így BC a betöltési cím.

Relokálás

SP-t elmentjük HL-be. HL-be a relokációs táblázat relatív címe kerül. Ehhez hozzáadva BC-t, megkapjuk a táblázat elejének abszolút címét. Ezt az értéket tesszük az SP-be. A veremmutató mindig a verem legalsó bájtjára mutat, a legkisebb címen lévő, még értékes bájtra, azaz az első szabad memóriahely fölé. Ezért a mi esetünkben SP-t a táblázat alsó bájtjára és nem az elé kell állítani. A POP HL utasítással így egyszerre tudjuk kivenni a táblázatból a következő szót és a táblázathoz tartozó mutatót, az SP-t a következő elemre állítani. Ellenőrizzük, hogy nem a táblázatot lezáró nulla értékhez érkeztünk-e. Ha igen, akkor a programot relokáltuk. Ha

```
;*****************************************************************
;*      önrelokáló minimális monitor        (MINIMONI)           *
;* Első meghíváskor relokálja magát.                             *
;* Minden további meghívási CALL xxxx                            *
;*                          DEFB brk                             *
;*                                    ahol xxxx a betöltési      *
;*                                    cím, brk a break point     *
;*                                    száma.                     *
;*    Kiírja a regisztereket és a brk számot, majd vár           *
;*    amíg a C vagy az A gombot meg nem nyomjuk.                 *
;*                                                               *
;*****************************************************************
        ORG     0                       ;a relokáció miatt
        LOAD    50000                   ;betöltési cím
JRUT    EQU     18H                     ;JR utasítás kódja
LASTK   EQU     23560                   ;SPECTRUM rendszerváltozó
CR      EQU     13                      ;kocsivissza
PAUSE   EQU     1F3DH                   ;SPECTRUM  ROM rutin
OFFSET  EQU     22                      ;eltolási érték a veremben
DARAB   EQU     12                      ;kiírandó sorok száma
LDHL    EQU     36H                     ;LD  HL,szám utasítás kódja
        JR      ENTER
REMAKE  LD      B,2                     ;két
SLFMCD  LD      (HL),LDHL               ;   bájt
OFFS    INC     HL                      ;   helyreállítása
        DJNZ    SLFMCD
ENTER   DI                              ;megszakítás letiltása
        LD      BC,0                    ;az egész tárban keres
        LD      A,LDHL                  ;LDHL bájtot
LOOK    CPIR                            ;   kettőt egymás
        CP      (HL)                    ;            után
        JR      NZ,LOOK                 ;ha a második nem egyezik
        LD      (HL),START-OFFS         ;ugró utasítás második bájtja
        DEC     HL                      ;ugró utasítás
        LD      (HL),JRUT               ;      első bájtja
        JR      REMAKE                  ;ugrás az 'átírt' bájtokra
START   DEC     HL                      ;HL-be
        DEC     HL                      ;   a kezdőcím
        DEC     HL                      ;         kerül
        DEC     HL                      ;   majd átesszük
        LD      B,H                     ;      BC-be
        LD      C,L                     ;   az értéket
        EXX                             ;HL'-be kerül
        LD      HL,0                    ;a SP
        ADD     HL,SP                   ;  régi értéke
        EXX
        LD      HL,LISTA                ;a relokacios tábla rel. címe
        ADD     HL,BC                   ;abszolut címe
        LD      SP,HL                   ;SP mutat a listára
CIKL    POP     HL                      ;köv. relokálandó szó rel.címe
        LD      A,H                     ;nulla
        OR      L                       ;   bájt jelzi
        JR      Z,BEFE                  ;a táblázat végét
        ADD     HL,BC                   ;a relokálandó ért. absz. címe
        LD      E,(HL)                  ;DE-be
        INC     HL                      ;a relokálandó
        LD      D,(HL)                  ;            cím
        EX      DE,HL                   ;DE-be

        ADD     HL,BC                   ;   DE+BC     (kezdőcím)
        EX      DE,HL                   ;      =relokált cím
        LD      (HL),D                  ;abszolut cím
        DEC     HL                      ;         vissza
        LD      (HL),E                  ;            a helyére
        JR      CIKL                    ;következő érték
BEFE    LD      H,B                     ;HL-be
        LD      L,C                     ;a betöltési cím
        INC     HL                      ; +1
        LD      (HL),MINMON-2           ;a JR utánra uj cím
                                        ;így legközelebb
                                        ;            a MINMON-ra ugrik
        EXX                             ;SP-be
        LD      SP,HL                   ;  vissza a régi érték
        LD      HL,2758H                ;SPECTRUM rendszer
        EXX                             ;   HL'-be
        EI                              ;megszakítás engedélyezés
        RET                             ;relokálás vége.
MINMON  PUSH    IY                      ;regiszterek
        PUSH    IX                      ;
        EX      AF,AF                   ;
        EXX                             ;
        PUSH    HL                      ;   elmentése
        PUSH    DE                      ;
        PUSH    BC                      ;
        PUSH    AF                      ;
        EX      AF,AF                   ;        a
        EXX                             ;
        PUSH    HL                      ;
        PUSH    DE                      ;
        PUSH    BC                      ;        verembe
        PUSH    AF                      ;
        LD      HL,0                    ;SP értéke a
        ADD     HL,SP                   ;      HL-be
        PUSH    HL                      ;            verembe
        LD      BC,OFFSET               ;HL-be az  SP
        ADD     HL,BC                   ;meghívás előtti értéke
        POP     BC                      ;BC-be SP értéke
        PUSH    HL                      ;verembe SP régi értéke
        LD      H,B                     ;HL-be a
        LD      L,C                     ;         SP értéke
        DEC     HL                      ;HL az elmentett
        DEC     HL                      ;   regiszterekre mutat
        LD      DE,SZOVEG               ;a kiírandó szöveg
```

```
CIM1    EQU     0-2                     ;          első relokálandó szó
        LD      C,DARAB                 ;a szövegek száma
KIR     CALL    WRT                     ;négy karakter kiírása
CIM2    EQU     0-2                     ;          második relokálandó szó
        INC     HL
        LD      A,(HL)                  ;a regiszterpár felső
        CALL    DIGIT                   ;bájtjának kiírása
CIM3    EQU     0-2                     ;          harmadik relokálandó szó
        DEC     HL                      ;a regiszterpár alsó
        LD      A,(HL)                  ;bájtjának
        CALL    DIGIT                   ;    kiírása
CIM4    EQU     0-2                     ;          negyedik relokálandó szó
        INC     HL                      ;HL léptetése a
        INC     HL                      ;következő regiszterpárra



        DEC     C
        JR      NZ,KIR                  ;ha van még
        CALL    WRT                     ;utolsó négy karakter kiírása
CIM5    EQU     0-2                     ;          ötödik relokálandó szó
        DEC     HL                      ;DE-be itt
        LD      D,(HL)                  ;    a PC
        DEC     HL                      ;    meghívás előtti
        LD      E,(HL)                  ;értéke  körül
        INC     DE                      ;PC-be PC+1
        LD      (HL),E                  ;vissza
        INC     HL                      ;    a
        LD      (HL),D                  ;verembe
        EX      DE,HL                   ;HL-be a PC
        DEC     HL                      ;meghívás előtti értéke
        LD      A,(HL)                  ;A-ba a BRK száma
        CALL    DIGIT                   ;    kiírás
CIM6    EQU     0-2                     ;          hatodik relokálandó szó
WAIT    CALL    KEY                     ;várakozás billentyűre
CIM7    EQU     0-2                     ;          hetedik relokálandó szó
        CP      'A'                     ;ha 'A' akkor a
        JR      Z,ABORT                 ;program megszakítása
        CP      'C'                     ;ha 'C' akkor a
        JR      Z,CONT                  ;program folytatása
        JR      WAIT                    ;nem jó billentyű
;*********************************
;a következő négy karakter kiírása
WRT     LD      A,CR                    ;soremelés
        CALL    PRNA                    ;kiírása
CIM8    EQU     0-2                     ;          nyolcadik relokálandó szó
        LD      B,4                     ;négy
WRT1    LD      A,(DE)                  ;  karakter
        CALL    PRNA                    ;    kiírása
CIM9    EQU     0-2                     ;          kilencedik relokálandó szó
        INC     DE                      ;következő karakter
        DJNZ    WRT1
        RET
;*********************************
;két hexadecimális számjegy kiírása
DIGIT   PUSH    AF                      ;a bájt a verembe
        SRL     A                       ;a felső
        SRL     A                       ;    félbájt
        SRL     A                       ;     az alsóba
        SRL     A                       ;          kerül
        CALL    HEX                     ;félbájt kiírása
CIM10   EQU     0-2                     ;          tizedik relokálandó szó
        POP     AF                      ;vissza a veremből
        AND     0FH                     ;a felső félbájt törlése
HEX     ADD     A,90H                   ;konverzió
        DAA                             ;     ASCII
        ADC     A,40H                   ;     kódra
        DAA                             ;rácsorog a kiíró rutinra
;*********************************
;egy karakter kiírása
PRNA    PUSH    AF                      ;regiszterek
        PUSH    BC                      ;    elmentése
        PUSH    DE                      ;       a
        PUSH    HL                      ;          verembe
        RST     10H                     ;SPECTRUM rendszer hívása




        POP     HL                      ;regiszterek
        POP     DE                      ;    visszatöltése
        POP     BC                      ;       a
        POP     AF                      ;          veremből
        RET
;*********************************
;a hívó program folytatása
CONT    POP     AF                      ;az eltett SP érték eldobása
        POP     AF                      ;
        POP     BC                      ;regiszterek
        POP     DE                      ;    ugyanabban a
        POP     HL                      ;       sorrendben vissza
        EX      AF,AF                   ;
        EXX
        POP     AF
        POP     BC
        POP     DE
        POP     HL
        EX      AF,AF
        EXX
        POP     IX                      ;a veremből PC
        POP     IY                      ;értéke eggyel nagyobb
        RET                             ;mint az eltett
;*********************************
;várakozás billentyűre
KEY     PUSH    BC                      ;regiszterek
        PUSH    DE                      ;
        PUSH    HL                      ;    elmentése
        LD      BC,0                    ;PAUSE argumentuma 0
        CALL    PAUSE                   ;várakozás
        LD      A,(LASTK)               ;A-ba a billentyű értéke
```

```
        POP     HL                      ;regiszterek
        POP     DE                      ;       értéke
        POP     BC                      ;          vissza
        RET
;*********************************
;a hívó program megszakítása
ABORT   EXX                             ;HL'-be
        LD      HL,2758H                ; a rendszernek
        EXX                             ;   megfelelő érték
        RST     8                       ;visszatérés a
        DB      0FFH                    ;   rendszerhez
;*********************************
;a kiírandó szövegek
SZOVEG  DB      "SP =AF =BC ="
        DB      "DE =HL =AF ="
        DB      "BC'=DE'=HL'="
        DB      "IX =IY =PC =BRK="
;*********************************
;relokációs táblázat
LISTA   DW      CIM1
        DW      CIM2
        DW      CIM3
        DW      CIM4
        DW      CIM5
        DW      CIM6
        DW      CIM7
        DW      CIM8



        DW      CIM9
        DW      CIM10
        DW      0                       ;lezáró nulla
        END
```

1. program

2. program

```
                        ;5.2 PROGRAM

                        ;MONS RELOKALAS

                                ORG     0D6D8H
D6D8 219013     LD6D8   LD      HL,1390H
D6DB 09                 ADD     HL,BC
D6DC 5E         LD6DC   LD      E,(HL)
D6DD 23                 INC     HL
D6DE 56                 LD      D,(HL)
D6DF 23                 INC     HL
D6E0 7A                 LD      A,D
D6E1 B3                 OR      E
D6E2 2811               JR      Z,LD6F5
D6E4 EB                 EX      DE,HL
D6E5 D5                 ADD     HL,BC
D6E6 D5                 PUSH    DE
D6E7 E5                 PUSH    HL
D6E8 5E                 LD      E,(HL)
D6E9 23                 INC     HL
D6EA 56                 LD      D,(HL)
D6EB EB                 EX      DE,HL
D6EC 09                 ADD     HL,BC
D6ED EB                 EX      DE,HL
D6EE E1                 POP     HL
D6EF 73                 LD      (HL),E
D6F0 23                 INC     HL
D6F1 72                 LD      (HL),D
D6F2 E1                 POP     HL
D6F3 18E7               JR      LD6DC
D6E5 C3F8D6     LD6F5   JP      LE22D
```

nem, akkor HL egy relokálandó bájtpárra mutat. Természetesen relatívan, ezért hozzáadjuk BC-t. DE-be az így kapott címről, a kisebb címről a kisebb helyiértékű (LSB), a nagyobb címről a nagyobb helyiértékű (MSB) bájtot töltjük, hozzáadjuk BC-t, és visszatesszük a helyére, majd vesszük a következő értéket a táblázatból.

Most már csak egy tisztázandó kérdés van a programmal kapcsolatban. Miért kellett letiltani a megszakítást? Ehhez először tisztázzuk, hogy mi is a megszakítás. Mindig valamilyen külső eszköz hozza létre. Amikor megszakítás történik, akkor a program futása megszakad, a programszámláló aktuális értéke bekerül a verembe, és valahol máshol folytatódik a futás. Ez a máshol a megszakító rutin, melynek az a feladata, hogy mindazt elvégezze, amiért a megszakítás történt, majd a regiszterek megváltoztatása nélkül visszaadja a vezérlést a megszakított programra.

A Spectrumban például 20 ms-onként történik megszakításkérés. Ezt egy, a processzor szempontjából külső eszköz, az ULA generálja. A megszakító rutinnak három feladata van: ha a processzor HALT állapotban van, akkor onnan kimozdítja; leolvassa a billentyűzetet; léptéti a belső órát.

A programozó számára ez azt jelenti, hogy a memóriában van három bájt, amely másodpercenként 50-szer automatikusan növekszik eggyel, van egy olyan, amelyik anélkül, hogy odatöltenénk bármit, tartalmazza az utoljára lenyomott billentyű kódját. Megszakítás azonban csak akkor történik a kérés hatására, ha a processzor IFF1 nevű jelzőbitje 1. Ez az IT (megszakítást) engedélyező bit. Számunkra most elég, ha azt mondom, hog ez a bit az EI utasítás (Enable Interrupt = megszakításengedélyezés) hatására 1, a DI (Disable Interrupt = megszakításstiltás) hatására 0 lesz. Ha tehát a processzor végrehajtott egy DI utasítást, akkor mindaddig, amíg EI utasítást nem hajt végre, hiába érkezik az INT lábra lefutó jel, azaz hiába történik megszakításkérés, a processzor nem vesz erről tudomást.

Amikor az a programrész fut, amelyik önmagát keresi a memóriában, akkor anélkül, hogy biztosak lehetnénk abban, hogy HL pontosan hová mutat, felülírjuk az általa megcímzett memóriarekesz tartalmát. Gondoljuk csak meg, mi történik akkor, ha azután, hogy átírtuk a bájtokat, de még azelőtt, hogy helyreállítottuk volna eredeti értéküket, történik egy megszakítás, és az átírt két bájt éppen a megszakító rutin része. Nagy valószínűséggel a gép lefagyását jelentené.

Ha a rövidebb, de rendszerfüggő megoldást választjuk, és meghívunk egy fix helyen lévő RET utasítást, akkor ha nincs letiltva a megszakítás, nem szükségszerűen, de megtörténhet, hogy mielőtt kivennénk a verem alól a PC értékét, a megszakító rutin tönkreteszi azt. Ennek egyébként 20 ms-onként létrejövő megszakítás esetén 0,00275% a valószínűsége, ami elég nagy, hiszen ha 1024-szer futtatjuk a programot, akkor már 94% a valószínűsége annak, hogy a katasztrófa egyszer bekövetkezik. Eladásra készült programok pedig ennél jóval többször is futnak.

Amikor az SP-t használtuk mutatóként a relokációs táblázathoz, akkor azért kellett letiltani a megszakítást, illetve letiltva hagyni, mert a verem alá, azaz a táblázatba belepiszkíthat a megszakítás. Igaz, hogy mindig az SP alá, a táblázatnak abba a részébe, amelyet már nem használunk többet, de a táblázat alatt ott van a program maga. A következő részben bővebben lesz szó megszakításról és megszakító rutinokról.

Egy kis fejtörő

A CP/M rendszer 12-es számú rutinja (ezt a számot a C regiszterben kell megadni, a belépési pont pedig minden rutinnál 5) a HL-ben megadja a rendszer verziószámát. Itt tehát HL foglalt, így a CP/M rendszerrutin nem fejeződhet be JP (HL) utasítással, hanem csak RET-tel! Cáfoljuk meg ezt az állítást!

VERHÁS PÉTER

Az UNIX operációs rendszer

Egy kis történelem

Az „operációs rendszer" mint szoftver a nagy számítóközpontokban született meg jó néhány évvel ezelőtt. Gyermekéveiben csupán eljárásgyűjtemény volt, amely a számítógép perifériáinak egyszerűbb kezelését szolgálta. Idővel nőttek az elvégzendő feladatok, velük tanult, okosodott az OS is. Életében nagy lépés, mondhatni az általános iskola első osztálya volt, amikor megtanult olvasni: megjelentek a parancsnyelvek. A programozónak programja futtatásakor már nem kellett bevonulnia a gépterembe kártyacsomagokat adogatni, szalagokat fűzögetni. Programja elé szúrt néhány kártyát, amelyek tudatták a rendszerrel, hogy milyen fordítót, milyen perifériákat, fájlokat, szalagokat fog majd használni. Ezt a kártyacsomagot leadta reggel, s ha szerencséje volt, már aznap este megkapta a lyukkártyákat készítő lányok és saját szintaktikus hibáinak jegyzékét.

A következő nagy lépés az interaktiv rendszerek megjelenése volt. A 60-as évek elején az Egyesült Államokban, a dartmouth-i egyetemen John G. Kemény és munkatársai kidolgozták az első interaktiv programozási nyelvet, a BASIC-et. A központi számítógéphez egy kisebb kommunikációs gépen keresztül különleges elektromos írógépeket, ún. teletype-okat kapcsoltak. A központi egység egyszerre több teletype-pal tartott kapcsolatot; amíg valaki felemelte ujját az egyik billentyűről, hogy másikat üssön le, esetleg 30 másik felhasználóval is foglalkozott. A teletype előtt ülő programozó úgy érezhette, egyedül vele foglalkozik a gép; a válaszidő néhány napról néhány másodpercre csökkent. Ez pozitív visszacsatolásként hatott: a rendszerek fejlődése is felgyorsult. Ebbe a hardver is „besegített": gyorsabb processzorok, nagyobb kapacitású operatív tárak, gyorsabb háttértárak, összetettebb segédáramkörök jelentek meg. A terminál mellett ülő felhasználó több programot is indíthatott, amelyek párhuzamosan, egymástól függetlenül futottak.

A gyors adatfeldolgozás iránti igények megnövekedtek, így megjelentek a multiprocesszoros rendszerek: több központi egység dolgozott közös felhasználóseregen és háttértáron. Aztán a párhuzamosan futó

programok (kezdetben csupán egyszerű flageken keresztül) kommunikálni kezdtek egymással. Az egyetlen probléma az maradt, hogy az operációs rendszerek készítői, a rendszerprogramozók ritkán találkoztak a felhasználókkal, nem ismerték eléggé igényeiket. Így a különböző csodanyelvek és rendszerek használata sokáig meglehetősen nehézkes maradt.

Fejlődtek a programozási nyelvek is: a bináris gépi kód és az assembler után megszületett az első magas szintű nyelv, a FORTRAN. Ma pedig már megszámolni is nehéz a különböző nyelvi rendszereket: PL/1, ALGOL, APL, Simula, BASIC, COBOL, GPSS, Lisp, CDL, BCPL, C, Pascal, Focal, FORTH, Logo, Prolog, Comal — hogy csak a legismertebbeket említsem. Megváltozott a programozók stílusa is, elterjedt az ún. strukturált programozás. Kernighan, Plauger, Ritchie, Dahl, Hoare, Thomson, Dijkstra — ezek a nevek ismerősen csengenek itthoni programozófülekben is.

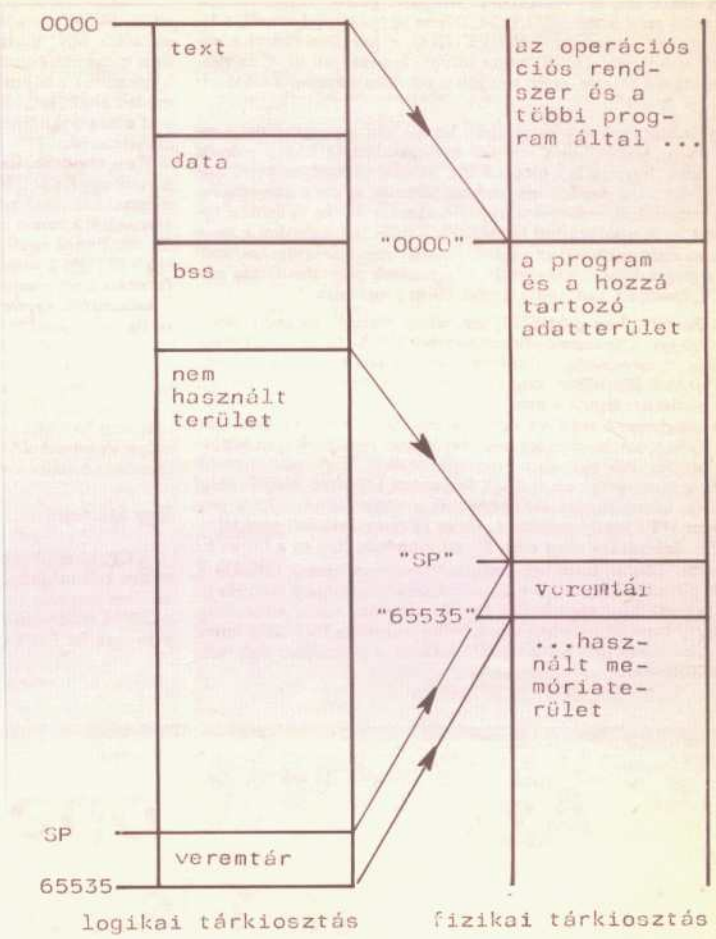
A hetvenes évek elején a BCPL-re és a B-nyelvre alapozva Ken Thomson és Dennis Ritchie elkészítették a C-programozási nyelv első fordítóprogramját és az UNIX operációs rendszer első változatát. Az UNIX és a C szorosan összetartoznak: az operációs rendszer elsősorban a C-fordítót támogatja, s magát az UNIX-ot is a C-nyelven írták. A C-t dicséri, hogy a mintegy tízezer sornyi program hatékonysága így is mindössze 5 százalékkal rosszabb, mint az assemblerben írt első változaté, de annál sokkal áttekinthetőbb, könnyebben javítható, módosítható.

Az UNIX és a C azért nem tűntek el a nyelvek és rendszerek egyre sűrűsödő erdejében, mert felhasználó-orientáltak. Tervezőik tanultak elődeik hibáiból, átvették a jónak tartott ötleteket, megkérdezték kollégáik véleményét is.

Egy évtized óta már az UNIX—7 is elterjedt, ami sikerét és a szoros — felülről — kompatibilitását mutatja. Alapelveit más rendszerekben is alkalmazzák. Bár maga az UNIX a PDP gépcsaládra készült, megtalálható más mini- és kisszámítógépeken is. Sőt, a fejlett 16 bites mikroprocesszorok elterjedésével már a személyi számítógépeken is megjelent. Hazánkban sajnos még a fehér hollónál is ritkább, ezért valószínű, hogy röpke tízéves lemaradásunkat e területen is sikerül megőriznünk . . .

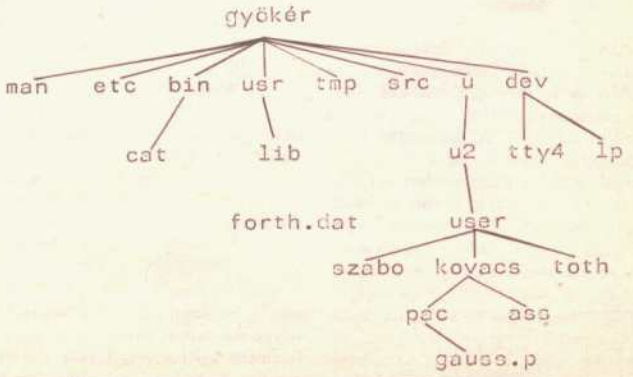
A következőkben két oldalról mutatom be az UNIX-ot. Az egyik részben magát az operációs rendszert, vagyis a programok és a rendszer kapcsolatát írom le, a másikban egy rövid példán keresztül az UNIX használatát ismertetem.

Arra nem is gondolhatok, hogy a teljes



1. ábra

2. ábra



rendszert sikerül megismertetnem az olvasókkal, csupán érdeklődésüket szeretném felkelteni egy modern, jól használható, magas szintű operációs rendszer iránt. Mivel az UNIX — egyes speciális funkciói kivételével — nem igényel komoly hardvertámogatást, látok némi reményt arra, hogy a közeljövőben egyre több magyar vagy más, szocialista gyártmányú számítógépen is találkozhatom vele (nem a PDP—11 kompatibilis gépekre gondolok).

A programok és az operációs rendszer kapcsolata

A UNIX többfelhasználós (multiuser) és többfeladatos (multitasking) operációs rendszer. Ez azt jelenti, hogy a rendszer képes több felhasználó fájljainak egymástól független kezelésére (olvasás, felülírás elleni védelmek), és a számítógépen párhuzamosan több program is futtatható; e programok kommunikálhatnak egymással. Mindenekelőtt szeretnék meghatározni néhány fontos alapfogalmat:

Rendszerhívás — az operációs rendszer valamely szubrutinjának hívása.

Megszakítás — az a folyamat, amikor egy külső jel hatására a számítógép felfüggeszti az éppen futó program végrehajtását, és rátér egy speciális kezelőprogramra, az ún. megszakításrutinra. Megszakításnak nevezik a fenti folyamatot kiváltó jelet is.

Fájl — valamilyen háttértárolón levő adatállomány, amely szöveget, gépi kódú programot vagy egyéb adatokat tartalmaz.

Katalógus — speciális fájl, amely más fájlok nevét és egyéb fontos adatait tartalmazza.

Terminál — billentyűzettel és képernyővel ellátott ki/bemeneti egység. Egy számítógéphez általában több terminált kapcsolnak, ezért a terminál egy belső processzor segítségével elvégzi a billentyűzet dekódolását, valamint a kapott karakterek kiírását, hogy a számítógépet tehermentesítse.

Processz — (folyamat, program), az operációs rendszer felügyelete alatt futó programok valamelyike.

A memória kiosztása

A számítógép memóriájában az egyszerre bent levő programok meglehetősen összevissza helyezkednek el. Az éppen futó program az MMU (Memory Management Unit) nevű hardversegédeszköz ügyes cím transzformációs mechanizmusán keresztül mégis úgy érzi, mintha a 0000-s címen kezdődne s teljes 64 kbájt terület állna rendelkezésére.

Egy tetszőleges program által használt 64 kbájtot öt fő részre, ún. szegmensekre lehet bontani. Az első rész maga a program, amelyet textszegmensnek is neveznek. Azért választották külön a hozzá tartozó adatterületektől, mert az MMU segítségével e szegmenst írásvédetté tudjuk tenni. Ez azt jelenti, hogy ha a program megpróbálja magát felülírni, „szegmenshatár-megsértés" megszakítást kap. Ilyenkor a program és az összes regiszter pillanatnyi állapota egy „core" nevű fájlba kerül, amit hibakeresésre használhatunk. Az adatterület célszerűségi okok miatt két részre van osztva: adat- és munkaterület- (bss) szegmensre. Az e fölött levő területet a program nem használja, s legfölül ott van a veremmemória, amely a 65 535 címen kezdődik, és szokás szerint lefelé növekszik. Az *1. ábrán* láthatjuk az imént leírt öt terület elhelyezkedését: a logikai memóriakiosztást, mellette pedig azt, hogyan néz ki ez a valóságban.

A nem használt memóriaterület tehát — takarékossági okokból — nem is létezik. Ha a program valamilyen hiba miatt mégis a nem megengedett területre hivatkozik, szintén „szegmenshatár-megsértés" megszakítást kap.

Mivel a veremtár méretét előre nem állapíthatjuk meg, a veremmutató által okozott szegmenshatár-megsértéseket az UNIX úgy orvosolja, hogy a program áthelyezgetésével — általában 640 bájtonként — növeli a veremterületet.

A feldolgozandó adatok mennyiségétől függően egy program használhat 200 vagy 20 000 bájt adatterületet. Fölösleges pazarlás lenne mindig 20 k-t lefoglalni — az UNIX-rendszerben ezért a *break* rendszerhívás segítségével futás közben is tetszőlegesen változtathatjuk a felhasznált adatterület méretét. Így gazdaságos, rugalmas memóriakihasználásra van lehetőségünk.

A fájlrendszer faszerkezete

Egy jól kiépített kisszámítógép felhasználóinak száma száznál is több lehet, így a rendszerben néhány ezer fájl tárolását és szervezését kell megoldani. A fájlokat tehát a kézben tarthatóság érdekében — részben típusuk, részben tulajdonosuk szerint — csoportokba kell rendezni. Az UNIX-rendszerben ezzel a módszerrel egy teljesen univerzális faszerkezetű fájlrendszert alakítottak ki.

A faszerkezet megvalósítása érdekében a tárolt fájloknak két típusuk van: minden fájl lehet közönséges *adatállomány*, illetve *katalógus*, amely újabb fájlokat tartalmaz. A kezelhetőséget persze meg kell őrizni, ezért az ilyen fában nem lehet túl sok emelet.

Egy átlagos UNIX-rendszer katalógusfájának részletét mutatja a *2. ábra*. Itt a *man* alkatalógus tartalmazza a használati utasításokat, a *bin* a gyakran használt segédprogramokat, az *usr* a ritkábban használt segédprogramokat, a különböző fordítókat és szubrutinkönyvtárakat, a *tmp* az átmeneti fájlokat; az *src*-ben a rendszerben levő majdnem minden program forrásprogramja van, az *etc*-ben a rendszer működéséhez szükséges programok és fájlok vannak, az *u*-ban pedig a felhasználók saját programjai és adatfájljai.

Ha ebben a fában egy adott fájlra szeretnénk hivatkozni, fel kell sorolnunk a hozzá vezető elágazási pontok nevét, azaz az alkatalógusokat, valahogy így:

/usr/lib/forth.dat
/u/u2/user/kovacs/pac/gauss.p
/bin/cat

Itt a "/" jel az elválasztó az egyes katalógusok neve között. Az első / előtt látható semmi a gyökérkatalógus neve: mivel egy rendszerben csak egy lehet belőle, külön azonosítóra nincs szüksége.

Az átlagos felhasználó általában csak a saját katalógusában dolgozgat, ezért ki tudja használni a rendszernek azt a szolgáltatását, hogy minden megadott fájlnév elé automatikusan hozzáilleszti a munkakatalógus nevét. Ez a munkakatalógus szükség esetén a *chdir* segédprogrammal megváltoztatható.

Az UNIX-rendszer első változatainak készítésekor még nem voltak olyan nagy kapacitású, gyors és olcsó háttértárak, amelyekre ez a rengeteg fájl egy darabban felfért volna. Részben ezért, részben pedig a könnyebb hordozhatóság kedvéért a fájlrendszert úgy tervezték meg, hogy a fa *bármely* elágazási pontjához hozzárendelhetünk egy háttértárolót. Az egyes készülékek (általában gyors merevlemezes diszkek) felszerelése a *mount* segédprogrammal, leszerelése pedig az *umont*-tal történik. A BME

Folyamatszabályozási Tanszékén levő egyik TPA 11/440-en futó UNIX-rendszerben például az *u* és az *src* alkatalógus egy-egy 27 Mbájtos nagy diszken van, a többi pedig 5 Mbájtos, SZM 5400-as bolgár kazettás diszkeken. Szükség esetén még mágnesszalagegységeket és 8"-os floppymeghajtókat is kapcsolhatnak a rendszerhez.

Ezt a felszerelhetőséget viszonylag egyszerűen valósították meg. Minden háttértároló-típushoz tartozik egy kezelőprogram, amely a perifériát 512 bájt hosszú sorszámozott blokkokra osztja. Az UNIX az így blokkokra osztott tárolóeszközön a hardvertől már teljesen függetlenül felépíti a fájlrendszert.

A „mount-olás" kissé már hardverbeavatkozás a rendszerbe, ezért ezt a parancsot csak az operátorok használhatják (őket az UNIX-ban superuser-nek, röviden su-nak nevezik). A saját katalógusán belül a rendszer faszerkezetét mindenki tetszés szerint építgetheti, az *mkdir* segédprogrammal új katalógusokat hozhat létre stb. De ez már a fájlkezeléshez tartozik.

Az univerzális fájlkezelés

Az elterjedt mikroszámítógépeket használók jelentős része talán még nem is találkozott a fájl fogalmával. A mikroszámítógépeken futó programok jellege még nem is teszi nélkülözhetetlenné használatukat, bár a kazettára vagy lemezre felvett program is egyfajta adatállomány. Ezért — tisztelet a kivételnek — a mikroszámítógépek állománykezelő műveletei meglehetősen szegényesek.

Egy BASIC-program javítása általában így zajlik: 1. Betöltjük a programot a lemezről vagy a kazettáról. 2. Elvégezzük a szükséges javításokat. 3. A kijavított változatot újra felvesszük.

Az UNIX-ban az 1. és 3. lépést a szövegszerkesztő program automatikusan elvégzi, sőt, ha gépelés közben áramkimaradás vagy más hiba miatt „elszáll" a gép, az addig beírt szöveget az editor már tárolta, legfeljebb az utolsó néhány betű vész el. De nem csak az editor, minden más program is a fájlokon keresztül érintkezik környezetével. A C-fordító egy szövegfájlból olvas, néhány közbenső fájl után a gépi kódú programot is egy fájlban tárolja. Ha el akarjuk indítani ezt a programot, be kell írnunk a fájl nevét.

Már az eddig leírtak is jól illusztrálják: ha van egy gyors háttértárolónk, megfelelő fájlkezeléssel igen kényelmessé válik mind a programozók, mind a felhasználók munkája.

Az UNIX-rendszerben egyféle, indexszekvenciális szervezésű adatállomány létezik. Egy indexszekvenciális fájlt sorban, bájtonként vagy szakaszonként végigolvashatunk (ez a szekvenciális fájlok jellemzője), vagy ide-oda lépkedhetünk benne (ez pedig a random fájlok tulajdonsága), esetleg felülírhatjuk egyes részeit. Ha elugrunk a fájl végére, még hozzá is fűzhetünk valamit; az egyetlen, egyébként érthető megszorítás: beszúrni nem lehet.

Az UNIX-ban megvalósított fájlkezelő műveletek:

1. *open:* közöljük az operációs rendszerrel, hogy melyik fájllal kívánunk dolgozni; ezt a fájl megnyitásának nevezik. Ilyenkor a rendszer létrehoz egy munkaterületet, amelyen a fájl állapotváltozóit, mutatóit tárolja. E munkaterület sorszáma az ún. állományleíró; a további műveletek során ezzel hivatkozhatunk a fájlra.

2. *close:* a fájllal befejeztük a munkát, a részére kijelölt munkaterületre már nincs szükség; ilyenkor a fájlt lezárjuk. A rendszer ekkor az esetleg még átmeneti pufferben levő információkat is a lemezre rögzíti.

3. *creat:* új adatállományok létrehozására szolgáló eljárás. Ha egy nem létező fájlt próbálunk meg opennel megnyitni, hibajelzést kapunk. A createljárás megvizsgálja, létezik-e már a megadott nevű fájl. Ha igen: törli, majd létrehozza a lemezen az új fájl fejlécét. Az új fájl állapotváltozóit adja vissza a B/K műveletek részére.

4. *write:* tetszőleges hosszúságú adatblokk írása egy megnyitott fájlba.

5. *read:* tetszőleges hosszúságú adatblokk olvasása egy megnyitott fájlból.

6. *readnl:* egy sor olvasása egy megnyitott fájlból.

7. *seek* vagy *lseek:* lehetőséget ad a fájlon belüli pozicionálásra. Az író/olvasó mutatót a fájl elejéhez, végéhez vagy az aktuális pozícióhoz képest tetszőleges értékre állíthatjuk be.

Az eddig leírtak ugyan nagyon kényelmes eszközök, de még nem különböznek lényegesen a hagyományos operációs rendszerek szolgáltatásaitól. Az UNIX tervezőinek egyik nagy ötlete volt, hogy a fájlkezelés fent leírt rendszerét kiterjesztették az egyébként eltérően kezelt B/K perifériákra is. A terminálok, a sornyomtatók, a soros adatátviteli vonalak, a mágneses háttértárolók fizikai blokkjai, sőt a rendszer memóriája is egy-egy speciális kezelőprogram segítségével mint fájlok érhetők el. Ezek a speciális fájlok a dev alkatalógusban szerepelnek, például:

/dev/tty4 — a 4-es terminál (teletype),
/dev/lp — a sornyomtató (lineprinter),
/dev/rrp0 — az egyik 27 Mbájtos lemezegység,
/dev/rrx0 — az egyik 8"-os floppymeghajtó.

Első látásra nem tűnik túl hasznosnak ez az uniformizálás, pedig van egy óriási előnye: a programoknak „belülről" nem kell megkülönböztetniük például a billentyűzetről vagy egy fájlból történő beolvasást. Így a terminál B/K-ra készített programot a megfelelő eszközzel belső módosítás nélkül rávehetjük, hogy például egy fájlból olvasson és a nyomtatóra írjon. Lehetőség van két program összekapcsolására is a *pipe* (cső) nevű B/K mechanizmussal.

Párhuzamosan futó programok

Mint már említettem, az UNIX többfeladatos operációs rendszer: a programok — a drága központi egység jobb kihasználása érdekében — egymást gyorsan váltogatva, a felhasználó számára azonban párhuzamosan futnak. A központi egység egy-egy processzel viszonylag kevés időt tölt; egy idő után a programállapotot elmenti az ún. processztáblázatba, majd a soron következő feladattal kezd el foglalkozni. Így minden processznek két állapota lehetséges: vagy fut, vagy arra vár, hogy futhasson (alszik). Nem az UNIX az első és az egyetlen rendszer, amely ezt a trükköt használja a felhasználók megtévesztésére. Ma is elterjedten alkalmaznak a PDP gépeken más rendszereket is: RSTS, RT—11, RSX—11 stb. Az UNIX időosztása azonban sokkal rugalmasabb ezeknél. Másodpercenként 50 vagy 60 alkalommal lefut a *scheduler* nevű program, s bizonyos tapasztalati szabályok szerint rontja a futó processz, javítja az alvó processzek prioritását. Ha valamelyik várakozó processz prioritása jobb lesz, mint az éppen futóé, helyet cserélnek: ez a folyamat az ún. feladatváltás.

Amikor egy processz befejezi működését, egy csomó adminisztrációt kell elvégezni: a még nyitott fájlok lezárása, a pufferek ürítése, törlés a processztáblából stb. Erre szolgál az *exit* eljárás. Az *exit* egyetlen paramétere egy állapotszó, amellyel jelzheti az operációs rendszernek, hogy rendben lefutott-e. Megállapodás szerint a 0 jelzi a hiba nélküli lefutást.

Ha egy program végtelen ciklusba került vagy egyéb kellemetlen dolgokat csinál, kénytelenek vagyunk „megölni". Erre a célra szolgál a *kill* rendszerhívás, illetve segédprogram, amely megszakítja a megadott azonosító számú processzt. Ezt az azonosító számot — ha szükség lehet rá —, a program indításakor közli az operációs rendszer. A *kill* parancsot természetesen mindenki csak a saját programjaira adhatja ki. Az UNIX-rendszerben a „seriff" szerepét a superuserek látják el: ők bármely futó programot leölhetnek.

Az előbb szerepelt egy új fogalom: a processz azonosító száma (process indentifier = PID). Az UNIX-on belül ez szolgál a

programok egyértelmű azonosítására, ezért az operációs rendszer indításától kezdve a leállásig nem lehet két egyforma azonosítójú processz. Minden futó processz lekérdezheti saját PID-jét a *getpid* rendszerhívással; ezt általában a biztosan különböző nevű átmeneti fájlok elnevezéséhez használják fel.

Egy többfeladatos rendszerben az egyik legfontosabb eljárás az új processz indítása. Az UNIX-ban erre a célra a *fork* (villa) nevű rendszerhívás szolgál. A *fork* első látásra meglehetősen haszontalan dolgot művel: lemásolja az őt meghívó processzt. Így a létrehozott új folyamat (a child: „gyerek"-processz) az eredetivel (a parent: „szülő"-processzel) teljesen megegyező lesz. Egyetlen apró dologban különböznek csak: a szülőprocessz megkapja gyerekének azonosító számát, míg a gyerekprocessz csak egy Ø-t kap. Ezt programelágazási feltételként alkalmazva elérhetjük, hogy lesz két különböző dolgot művelő programunk — ez azonban még kevés. Jó lenne, ha egy processz megkérhetné a rendszert: „Légy szíves, hajtsd végre a benőke.out nevű bináris fájlban tárolt programot!"

Majdnem pontosan ezt a funkciót valósítja meg az *exec* rendszerhívás. Végrehajtja a megjelölt programot, de közben felülírja és kitörli az exec-et meghívó processzt. Emiatt egy új folyamatot csak két rendszerhívás, a fork és az exec segítségével indíthatunk el: programunk először megduplázza önmagát (osztódik), majd általában a gyerekprocessz önfeláldozóan meghívja az exec-et. Ezt a nehézkesnek tűnő két részre vágást az indokolja, hogy az esetek többségében a fork és az exec között szükség van egy kis adminisztrációra, amit egybeépített fork—exec esetén csak bonyolultabban lehetne elvégezni.

Ha egy futó program szükségét érzi, saját aktuális prioritását is megváltoztathatja. De ha ezt mindenki tetszőlegesen megtehetné, az egyes processzek nyakra-főre javítgatnák saját prioritásukat; hamar kiérnénk a számábrázolási tartományból. Ennek elkerülésére az átlagos felhasználó programjai saját prioritásukat a többiek javára csak ronthatják. Ha valaki elmegy fél órára, nyugodtan elviseli, hogy programja tíz helyett húsz perc alatt fut le, ezzel kicsit gyorsítva a többiek futását. A superuserek általában fontos dolgokat csinálnak, jogaikkal is csak előre élnek, vissza soha. Ezért nekik többek között az is megengedett, hogy prioritásukat javítsák, sőt, állandóan a maximumon tartsák; ilyenkor aztán a többi program betűhöz sem jut.

Ébresztőóraként használható a *sleep* eljárás, amellyel a program végrehajtását másodpercekben megadható időre felfüggeszthetjük.

Mint már a fork hívásnál említettem, az osztódott processzek egyikét szülőnek, a másikat gyereknek nevezik. Mivel a szülő is végrehajthat új forkokat, a gyerekek is, valóságos családfát képezhetnek ezzel. Egy ilyen családfában előfordulnak olyan érdekességek, hogy a szülő egy részletszámítás elvégzését egyik gyerekére bízza, amíg ő egy másikkal bajlódik. Ilyen esetekben lehet arra szükség, hogy a szülőprocessz megtudja: fut-e még a gyerek? Erre szolgál a *wait* rendszerhívás: felfüggeszti a szülő végrehajtását, amíg valamelyik gyereke befejezi működését. A szülő ekkor megkapja a lefutott processz PID-jét, ebből megállapíthatja, hogy a várt program fejeződött-e be, valamint az állapotszót, amit az exit-tel küldött el neki a néhai.

A többfelhasználós rendszer

Egy nagy teljesítményű, gyors számítógép a hozzá tartozó perifériákkal igen drága jószág. Ha egy kisvállalat szeretné számítógépesíteni ügyvitelét, emellett még sok programra és személyzetre is szüksége volna, s a drágán vásárolt rendszer csak napi egy-két órát futna — szóval kihasználatlan maradna. Ezért jöttek létre a számítóközpontok, amelyek vagy egyetemeken, főiskolákon szolgálják az oktatást, vagy bérbe adják gépük minden másodpercét.

Kezdetben vala a kötegelt feldolgozás, amelyet a bevezetőben már említett hátrányai miatt senki sem tartott tökéletesnek. A párbeszédes rendszerek elterjedésével még inkább előkerültek a többfelhasználós rendszer előnyei: a számítóközpont jelentősebb beruházásokat hajthatott végre, mert több felhasználó, nagyobb tőke állt mögötte. Az új programokat, eszközöket pedig minden arra jogosult használhatja.

Meg kell azonban tanulni az együttélés szabályait is. Nincs jogom az UNIX-rendszerben mások fájljainak felülírására, ha a tulajdonos titkolózó természetű, még olvasására sem. Ezeket a jogokat a fájlok tulajdonosai szabhatják meg a *chmod* rendszerhívás, illetve segédprogram használatával. A superuserek természetesen minden fájlt olvashatnak, sőt még azok tulajdonosát is megváltoztathatják, átruházva az ezzel járó jogokat.

Miben jelent felügyeletet az UNIX?

Sok helyen használják azt a kifejezést, hogy „az operációs rendszer felügyelete alatt" futnak a programok, néhány helyen talán jogtalanul. Én a magam részéről a CP/M-rendszert még nagy jóindulattal sem merném felügyeletnek nevezni, de nem is ezzel a céllal hozták létre.

Az UNIX azonban már nem csak kiszolgálja a programokat, vigyáz is rájuk. Az operációs rendszeren belül futó programok a külvilággal csak a rendszerhívásokon keresztül érintkezhetnek. Ezek a hívások zárt rendszert alkotnak, amelyeken keresztül (majdnem) minden értelmes dolgot el lehet érni, ezért nincs szükség a program és a környezet számára egyaránt veszélyes „ablakokra". Ha egy program hibás adatok, külső beavatkozás vagy tervezési hiba miatt a környezetre veszélyes dolgokat kezd csinálni, ezek nagy valószínűséggel felkeltik az operációs rendszer figyelmét és programmegszakításhoz vezetnek. A programmegszakításoknak kb. tucatnyi oka lehet; a szegmenshatár megsértésén kívül címzési hiba, illegális utasítás, lebegőpontos hiba, megszakítás a terminálról stb. Nagyobb programok felkészülhetnek az ilyen kemény hibák kezelésére is, ezért a *signal* rendszerhívás segítségével elfoghatják vagy figyelmen kívül hagyhatják e megszakításokat.

Ha megpróbálunk írásra megnyitni egy olyan fájlt, amelyre nincs ilyen jogunk, az *open* rendszerhívás assemblerben bebillentett carry bittel, C-ben −1 állományleíróval tér vissza. Hasonló hibajelzést kaphatunk minden rendszerhívástól, ha az felkészült a hibás paraméterek fogadására. Az ilyen jelzések feldolgozása általában egyszerű, megfelelő programozással mégis nagyon intelligensnek és figyelmesnek tűnő programokat készíthetünk segítségükkel.

Az UNIX sok szolgáltatását már csak megemlíteni tudom. Ilyenek a végrehajtási idő mérése, a futásidő alatti keresztmetszet készítése a programról, a dátum és az óra lekérdezése stb. Az egyéb speciális funkciókat igen ritkán használják, ezért nem érdemes rájuk kitérni.

A felhasználó és az operációs rendszer kapcsolata

Az UNIX eddig megismert része: az operációs rendszer eljárásai csupán szubrutinok, amelyeket programokba kell beépíteni. A legfontosabb ilyen programok azok a parancsok, amelyek a kényelmes és hatékony használathoz nélkülözhetetlen kapcsolatot biztosítják a felhasználók és a rendszer között. Ezeket az ún. segédprogramokat az UNIX tervezői és a korábbi verziók felhasználói készítették el — természetesen C-nyelven. Ez a néhány száz(!) program persze már sok helyet foglal el, ezért általában mágneslemezen vannak a már említett bin, usr és etc katalógusokban, és csak akkor töltődnek be a tárba, ha valaki használni akarja őket.

Az UNIX készítőinek filozófiája: minél ritkábban legyen szükség arra, hogy egy apró feladat megoldása — mint például egy fájl nagybetűsítése — öt-tíz perc időveszteséget okozzon egy programozónak. Ezért

készítettek ilyen sok, viszonylag egyszerű és univerzális segédprogramot, mint például az előbbi feladat megoldására alkalmas *ucase*-t vagy párját, a kisbetűsítő *lcase*-t.

A fontosabb segédprogramokat a *táblázat* tartalmazza, használatukat majd példán láthatjuk.

A két legfontosabb segédprogram a *login* és a *shell*. A login a be- és kijelentkezéseket intézi, valamint minden felhasználó részére elindít egy parancsfeldolgozót. A parancsfeldolgozó kapta a shell (=kagyló) nevet. Hogy mi mindent tud, az a példából is látszani fog, de mielőtt a lényegre térnék, egy apróság: az alábbiak a BME már említett TPA 11/440-én futó UNIX „tájszólásra" igazak, amely a helyi fejlesztéseknek köszönhetően az átlagos UNIX—V6-nál kicsivel több segédprogramot tartalmaz.

A feladat egy hatismeretlenes lineáris egyenletrendszer megoldása. Szeretnék minél előbb végezni vele, ezért két előadás közötti szünetben betérek a szomszédos R épületben levő számítógéphez.

A képernyős terminálon a "hívónév:" felirat olvasható, mögötte ott villog a kurzor. Begépelem a hívónevemet; devil, majd várok. Pár másodperc, amíg a login program megkeresi a felhasználók listáján a devil kezdetű bejegyzést, s máris megjelenik a "Jelszó:" felirat. A jelszavamat természetesen nem árulom el, de ha valaki az ujjaim mozgásáról mégis leolvasná, a *passwd* programmal bármikor megváltoztathatom. A begépelt jelszó egyezik a korábban tárolttal, ezzel be is jelentkeztem a rendszerbe.

A shell kiírja a promptot (ez általában egy unalmas %-jel, de a *set* programmal tetszőlegesen módosítható), amellyel jelzi, hogy kész a következő parancssor végrehajtására. A parancssorok első elemének mindig egy programot tartalmazó fájl nevének kell lennie. A legegyszerűbb esetben a shell nem tesz mást, csak betölti és elinditja a megadott nevű programot. % ls

Az *ls* segédprogrammal a lemezen tárolt fájljaim nevét tudom kilistázni. Mivel az ls soronként csak egy nevet ír ki, a terminálra pedig tizenhat sor fér ki, a lista eleje hamar elszaladt. A megoldás az *mcol* program, amely a billentyűzetről olvasott szöveget több hasábban írja ki a képernyőre. A shell pedig képes arra, hogy az ls kimenetét a korábban már említett pipe-pal az mcol bemenetére kapcsolja. Már jelentkezik az univerzális fájlkezelés előnye! % ls|mcol

Az "|" jel utasítja a shell-t a csővezeték létrehozására. Természetesen „hosszabb" pipe-ot is készíthetünk, ennek egyes pontjaiból a *tee* segédprogrammal mintát is vehetünk.

A többhasábos listázásra gyakran lehet szükség, ezért érdemes az *alias*-szal rövidíteni az előbbi parancslâncot (*3. ábra*, 3. sor). A továbbiakban az *l* önálló parancsként fog viselkedni. Aki ismeri a CP/M-rendszert, most a submit-re gondolhat, bár annak közelebbi rokona, az ún. parancsfájl is létezik az UNIX-ban.

A chdir-rel (4. sor) saját kis katalógusfámban most egy szinttel lejjebb, a cee nevű alkatalógusba léptem, ahol C-nyelvű programjaimat tárolom. A ";" két független parancs közötti elválasztójel, az l pedig az előbb definiált listázó parancs.

A lináris egyenletrendszer megoldásának legcélszerűbb módszere a Gauss-elimináció. Ennek programját már korábban megírtam, most csak le kell fordítani (5. sor). A "−0" kapcsolóval arra kértem meg a C-fordítót, hogy a tárgyprogramra eressze rá az optimalizáló menetet is. Így a fordítás ugyan kicsit tovább tart, de a gépi kódú program rövidebb és gyorsabb lesz. Ez most nem fog sokat számítani, hosszabb programoknál azonban már egész nagy lehet a különbség. A fordító már végzett is, elkészült az a.out nevű bináris fájl. A C-fordító mindig a.out-ot készít, ezért jobb megszokni, hogy ezt rögtön át is nevezzük (6. sor).

A gaussbin program az együtthatókat a billentyűzetről olvassa, az eredményeket pedig a képernyőre nyomtatja ki. Jó lenne egy maradandó dokumentáció a megoldásról. Ennek semmi akadálya, a shell segítségével a képernyőre kerülő szöveget tetszőleges fájlba irányíthatjuk át (7. sor). A „kacsacsőrrel" most az ún. standard outputot vezettük az eredm nevű fájlba. Ha ez a fájl már korábban is létezett volna, a shell kíméletlenül törli.

A dupla kacsacsőr felhasználásával egy korábban létező fájl végéhez fűzhetünk újabb adatokat (8. sor). A "»" jellel a sorszámozott programlista a végeredmények mögé került, így egész szép dokumentációt kaptam, amit már csak ki kell nyomtatni.

A (9. sor) *wc* szerint a fájlom 71 sor hosszú, egy lapra pedig csak 66 sor fér ki. Kár volna azért az öt sorért új lapot kezdeni! Az eredm leghosszabb sora is csak 54 karakter hosszú (10. sor), így a 132 oszlopos nyomtatóval két hasábban is kiírhatjuk (11. sor).

Nehogy most bárki azt gondolja, hogy mellettem ott kattog a nyomtató! Az a gépteremben van, alszik. Ha elegendő nyomtatni való gyűlt össze, bekapcsolják. Akkor öt perc alatt ledarál egy nagy kupac papírt, majd visszafekszik aludni. Addig, amíg újra fel nem kel, a nyomtatnivalók megfelelő helyen várakoznak sorukra.

Van még pár percem, ezalatt pár soros üzenetet küldök egy tankörtársamnak, akinek hívóneve yozi (12. sor). Ebben közlöm vele, hogy a gauss.c működik, ha akarja, használhatja. A levél szövegét a billentyűzetről az end-of fájlnak megfelelő strl-D-val zárom le. Ha yozi legközelebb bejelentkezik, a rendszer közölni fogja vele: „Leveled van!"

Az *rm* program segítségével (13. sor) kitörlöm a most már fölösleges eredm és gaussbin fájlokat. Hogy bárki megláthassa: nem bonyolult dolog egy ilyen segédprogram, a *4. ábrán* egy olyan C-nyelvű programot közlök, amely az rm alapfunkcióját már képes ellátni. A tényleges rm ennél kicsit okosabb.

Mielőtt kilépek a rendszerből, megnézem, elkészült-e már a nyomtatás (14. sor).

„Nyomtatásra várnak:
jutka a tty4-ről
devil a tty7-ről
system a tty3-ról
wolf a ttyg-ről"

Ugye nem kell hozzá magyarázat? Kinyomtatom még ezt a tucatnyi parancssort is a *history* segítségével, majd kilépek, mert lassan kezdődik a következő előadás (15. sor). Szólok még egy operátornak, hogy szükségem lenne a nyomtatmányomra, és már megyek is. A *logout* intézi el a kijelentkezéssel kapcsolatos adminisztrációkat.

Az egyenletrendszer megoldása a számítógéppel összesen kevesebb, mint öt percig tartott, így időt és nem kevés gyalogszámítást takaritottam meg.

3. ábra. A shell által feldolgozott parancssorok

```
 1 ls
 2 ls mcol
 3 alias l „ls mcol"
 4 chdir cee;l
 5 cc-Ø gauss. c;l
 6 mv a.out gaussbin
 7 gaussbin>eredm
 8 num gauss.c>>eredm
 9 wc eredm
1Ø maxlin eredm
11 opr −2 eredm
12 mail yozi
13 rm eredm gaussbin
14 lpl
15 history opr; logout
```

4. ábra. Az rm segédprogram vázlata

```
main (argc,argv)
  int argc;
  char*argv[ ];
  {
      register i;
      for (i=1; i<argc; i++)
          if (unlink (argv [i]))
              printf ("%s: not removed\n", argv [i]);
  }

argc:     a shell által az rm-nek átadott paraméterek száma.
argv:     a paramétereket (sztringeket) tartalmazó vektor.
register: az i változó speciális tárolásának deklarációja.
unlink:   „fájl törlése" rendszerhívás a UNIX-ban. Ha a törlés sikerült, Ø-t ad vissza, ha nem, -1-et.
printf:   a formátumozott nyomtatás a C nyelvben.
```

Az UNIX fontosabb programjai

alias — gyakran ismétlődő parancssorozatok rövidítése
bas — egyszerű BASIC-interpreter
chdir — munkakatalógus megváltoztatása
cc — a C-nyelv fordítóprogramja
cp — fájl másolása
dc — zsebszámológép-program
e — profi szövegszerkesztő
fc — FORTRAN fordítóprogram
grep — szó keresése egy fájlban
history — a shell által eddig feldolgozott parancssorok kilistázása
kill — megszakítás küldése egy processznek
lisp — a LISP nyelv interpretere
ls — fájlok listázása
maxlin — fájl leghosszabb sorának megkeresése
mv — fájl átnevezése vagy mozgatása
mail — levelezés
num — fájl sorainak számozása
opr — fájlok nyomtatása
pc — a Pascal-nyelv fordítóprogramja
passwd — jelszó definiálása
rc — ratfor fordítóprogram
set — a parancsfeldolgozó fontosabb paramétereinek beállítása
sh — a shell
rm — fájlok törlése
troff — az egyik szövegformázó program
uniq — ismétlődő sorok keresése egy fájlban
wc — megszámolja egy fájl szavait és sorait
yacc — fordítóprogram-generátor program

A közben felhasznált segédprogramok ismertetése kissé talán rövid és gyors volt, de mint már említettem, cikkem célja csupán ismertetés és kedvcsinálás, nem oktatás.

MÉSZÁROS LÁSZLÓ

Szorzás és osztás assembly programokban

Már kisebb assembly programokban is előfordul, hogy valamilyen számítást kell elvégeznünk. Emiatt BASIC nyelvű részt építeni egy programba nehézkes és nem is túl elegáns. Ha ismerjük pontos használatukat, meghívhatjuk az interpreter beépített aritmetikai eljárásait, de ezek legtöbbször fölöslegesen pontosak — emiatt lassúak —, esetleg nem is használhatók. Ilyenkor van szükség saját aritmetikai rutinokra.

Amíg csak összeadásról vagy kivonásról van szó, nincs gond, hiszen ezeket egyszerűen felépíthetjük a mikroprocesszor utasításaiból. Problémás viszont a szorzás, még inkább az osztás megvalósítása. E két utóbbi művelethez adok most egy kis segítséget.

A bináris szorzás

A kettes számrendszerbeli szorzás alapelve megegyezik az ismert decimális szorzáséval, csupán a kettes szorzótábla jóval egyszerűbb: ha az első tényező 1, az eredmény a második tényező lesz, míg ha 0, az eredmény is nulla.

A módszer tehát: „végiggyalogolunk" az első tényező minden egyes bitjén. Ha a vizsgált i. bit 1, az eredményhez a megfelelő helyi értéken (i-vel balra léptetve, azaz 2^i-vel szorozva) hozzáadjuk a másik tényezőt. Ha a vizsgált bit O, 0-t adunk az eredményhez, vagyis nem csinálunk semmit. Az *1. ábrán* egy 4 bites, „kézzel" elvégzett szorzás, a *2. ábrán* pedig az előbbi gondolatmenet folyamatábrája látható.

A módszer egy gyakorlati megvalósítását mutatja a *3. ábra;* ennek egyetlen lépését fogjuk most megvizsgálni.

Az X tényezőt egy bittel balra tolva annak legnagyobb helyi értékű bitje a carry flagbe kerül. Amennyiben ez a bit 1, az eredményhez hozzáadjuk az Y tényezőt. Csökkentjük a számlálót; ha még nem érte el a 0-t, akkor pillanatnyi eredményünket eggyel balra toljuk, majd az X tényező következő bitjének vizsgálata jön. Fejben is ellenőrizhető, hogy a ciklus n—1-szer fog lefutni; így az X tényező legfelső (n—1-edik) bitjéből képzett részszorzat éppen n—1 db balra tolást szenved el, azaz pontosan a helyére kerül.

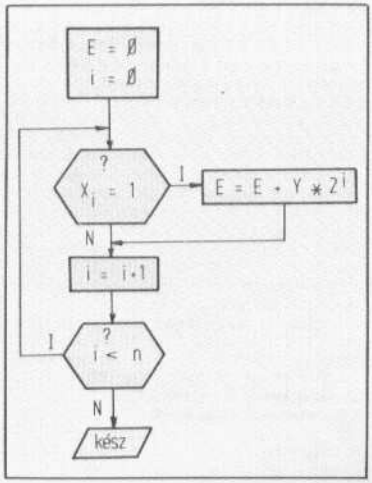
Ha nagyobb pontosságú, például 48 vagy 64 bites szorzást végzünk, a sebesség növelése érdekében a tervezéskor érdemes figyelembe venni, hogy 8 bitléptetés egyenlő 1 bájt léptetésével. Ügyes programozással akárhány bájtos szorzást készíthetünk úgy, hogy az operandusokat összesen 7 bittel toljuk el!

A *4. ábrán* Z80 és PDP—11 assembly nyelven megírva látható az előbbi folya-

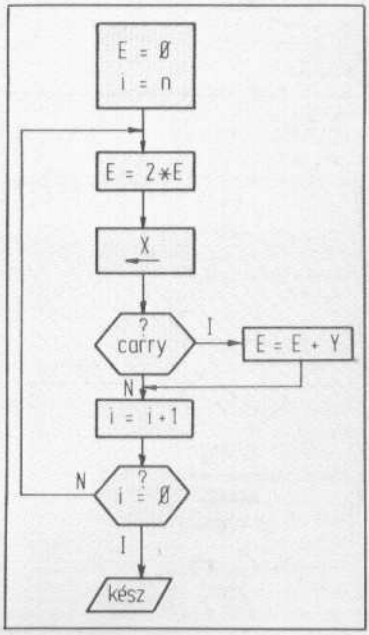
```
   1001·1101          = 9·13=117
   1001
  0000
 1001
1001
01110101        =64+32+16+4+1=117
```

1. ábra

2. ábra



3. ábra

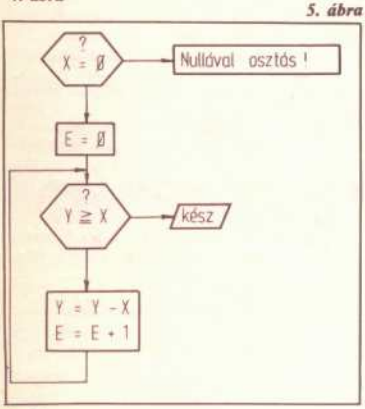


```
; BC_HL = BC*DE előjel nélküli szorzás
; Z80 assembly nyelvű változat
;
mull6:  ld      hl,0
        ld      a,16
loop:   add     hl, hl
        rl      c
        rl      b
        jr      nc, skip
        add     hl, de
        jr      nc, skip        ;***
        inc     bc              ;***
skip:   dec     a
        jr      nz, loop
        ret
; a 16 bites végeredményt adó változat
; gyorsítható az a és b regiszter szerepé-
; nek felcserélésével.
;
```

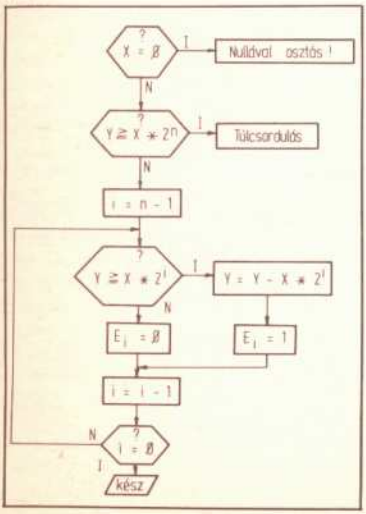
```
; rl_r0 = rl*r2 előjel nélküli szorzás
; pdp-11 assembly nyelvű változat
;
mull6:  clr     r0
        mov     #16.,r3
loop:   add     r0,r0
        rol     rl
;
bcc     skip
        add     r2,r0
        bcc     skip            ;***
        inc     rl              ;***
skip:   sob     r3,loop
;
        rst     pc
; a pdp-11 assemblyben az első operandus
; a forrás, a második a cél minden két-
; operandusú műveletnél. A „sob" nagyon
; hasonlít a Z80 „djnz"-jéhez.
```

4. ábra

5. ábra



6. ábra



matábra alapján készült szorzó szubrutin, amely két 16 bites előjel nélküli szám 32 bites szorzatát képezi.

Mindkét program tartalmazza azt a kis trükköt, hogy a fölfelé kilépkedő X tényező helyére alulról óvatosan bejön a 32 bites végeredmény felső 16 bitje. A Z80 változat paraméterei: az E eredmény alsó 16 bitje a HL regiszterpárban, az eredmény felső 16 bitje az X operandus helyén, a BC regiszterpárban lesz. Az Y tényező marad a DE regiszterpárban.

A PDP—11 egy 16 bites miniszámítógép; hat általános célú regisztere is 16 bit hosszú. Mivel a PDP-11 — igaz, előjelesen — ismeri a MUL utasítást, az összehasonlítás célja csupán a Z80 utasításkészletének dicsérete.

Ha valaki megelégszik 16 bites végeredménnyel is, a ***-gal megjelölt utasításokat el is hagyhatja.

A bináris osztás

Az osztás elvben egyszerűbb művelet, megvalósításában azonban problémák merülnek fel. Mert hogyan is osztunk? Megnézzük, hogy az osztót hányszor tudjuk kivonni az osztandóból, hogy még pozitív számot kapjunk. Az utolsó kivonás végeredménye a maradék, a kivonások száma pedig a hányados *(5. ábra)*. A módszer tökéletes, egy apró hátrányától eltekintve: sokáig tart. Vegyük például egy 32 bites szám osztását egy 16 bitessel; legyen az osztandó értéke a MAXINT: $2^{32}-1$ (négymilliárd körüli szám), az osztóé pedig 1. Ki várná meg a végeredményt, amely egyébként túlcsordulás lenne?

Segítsünk magunkon! Próbáljunk elvégezni egy részosztást az osztó kétszeresével, feleannyi lépésben: így kapunk egy részhányadost és egy részmaradékot. Ha a részhányadost kettővel szorozzuk, és hozzáadjuk a részmaradék és az eredeti osztó hányadosát, megkapjuk a tényleges hányadost. A részmaradék és az eredeti osztó hányadosa — belátható — csak 0 vagy 1 lehet. Bináris szaga van a dolognak; nézzük meg közelebbről.

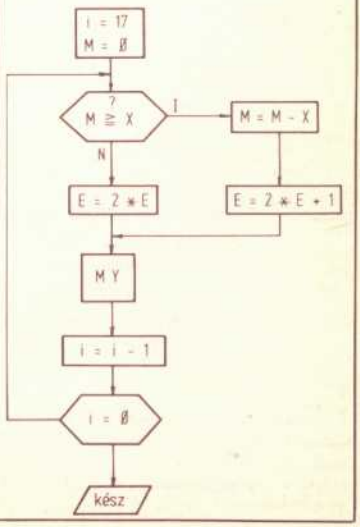
Ha az osztandó nagyobb, mint a hányados 2^{16}-szorosa, túlcsordulás lesz, mert a hányados nem fér el 16 biten. Tételezzük most fel, hogy nem keletkezik túlcsordulás. Osszuk el az osztandót az osztó 2^{15}-szörösével. A kapott részhányados értéke 0 vagy 1; ezt megszorozva 2^{15}-tel, a végleges hányados 15 legfelső bitjét kaptuk. A részmaradék kisebb, mint az osztó 2^{15}-szöröse. Osszuk el ezt az osztó 2^{14}-szeresével: a 14. bitet kaptuk. És így tovább a végeredményig *(6. ábra)*.

Z80 assemblyben a 32/16 bites osztást nem túl egyszerű megírni, mert kevés a három regiszterpár. Viszonylag egyszerű, rövid, érthető, az esetek többségében jól alkalmazható a 16/16 bites osztás, amelynek listája és működési diagramja a *7. ábrán* látható.

Az M paraméter, amely végül a maradékot fogja tartalmazni: a HL regiszterpár. Az osztandó Y a DE; ide fog alulról becsúszni a hányados is. Az osztó X-et a BC regiszterpár tartalmazza.

7. ábra

```
div16:  ld      hl,0
        ld      a,17        ; n+1
        or      a           ; a carry törlése
loop:   adc     hl,hl
        sbc     hl,bc
        jr      nc,count    ; ha sikerült
                            kivonni, számol
        add     hl, bc      ; ha nem, HLT
                            visszaállítja . . .
        scf                 ; . . . és törli
                            a carry-t
count:  ccf
        rl      e           ; 2-vel szorozza
                            DE-t,
        rl      d           és hozzáadja
                            a carry-t
        dec     a           ; a dec nem bántja
                            a carry-t!
        jr      nz, loop
        ret
```



COMMODORE 64

Verem

Az alábbi program egy LIFO (last in — first out) elven működő vermet hoz létre. A verem a BASIC és a KERNAL ROM alatt helyezkedik el, így BASIC programterületet nem foglal. A verem ilyen szervezése mellett 2731 valós vagy 5461 egész szám tárolható.

A program használata

— Használat előtt a verem inicializálása a SYS 12*4096 utasítással történik.

— PUSH-funkció: SYS 12*4096+3, [változó]. A program megkülönbözteti az egész és valós változókat. Az egész változók a veremben csak három bájtot foglalnak, a valós változók hatot. Kiolvasáskor nincs automatikus típuskonverzió, tehát ha a veremben egész érték következik, akkor az csak egész változóba tölthető. Szövegek verembe helyezésére nincs lehetőség.

— POP-funkció: SYS 12*4096+6, [változó]

A program leírása

1060—1130: A címkék definiálása. A $FE—$FF cím a veremmutató. A $47—$48 címet az ADRESS rendszerrutin állítja a BASIC szövegben következő változó értékének kezdőcímére. A $0E és a $0D címeket az ADRESS rutin a változó típusának jelzésére használja.

1150 : A program a $C000 címen kezdődik.

1170—1190: Ugrásvektorok.

1210—1220: A veremkezelő rutin rendszerváltozója. A rutin futásának idején a veremmutató értékét tartalmazza. A hibakezeléshez szükséges, hogy a verem tartalmát hiba esetén rekonstruálja.

1240—1280: Az inicializáló rutin. Beállítja a veremmutató kezdőértékét és az inicializálás flaget. Ez utóbbi értéket mind a PUSH, mind a POP rutin megvizsgálja, és ha nem történt inicializálás, hibaüzenetet küld.

1320—1960: A PUSH-funkciót megvalósító rutin.

1320—1350: Megvizsgálja az inicializálás flaget. Ha még nem történt inicializálás, akkor az ötös hibaüzenetre ugrik ("NO INITIALISED").

1360 : Meghívja az ELRAK szubrutint. Ez a rutin a veremmutató tartalmát az EL címre menti.

1370—1380: A változó címének és típusának meghatározása az ADRESS rutin segítségével.
$47—$48 = a változó értékének címe
$0D = #$FF, ha a változó szöveg
$0E = #$80, ha a változó egész

1390—1420: Megvizsgálja, hogy szöveges változó-e. Ha igen, akkor a megfelelő hibaágra ugrik ("STRING NOT EXAMPLE").

1430 : Az INTELL szubrutinra ugrik. Ez a rutin a $FC címen visszaadja a verembe másolandó bájtok számát.

1440—1600: A verembe másoló programrész. Az X regiszter a ciklus számláló. Először kikapcsolja a ROM-ot, átmásol egy bájtot és bekapcsolja a ROM-ot. A változó értékére mutató címkét megnöveli eggyel, a veremmutatót eggyel csökkenti (VMCSOKK rutin). Megnöveli a ciklusszámlálót, ellenőrzi, hogy egyenlő-e az INTELL rutin által beállított értékkel. Ha nem egyenlő, akkor folytatja a másolást.

1610—1700: A verembe írja a változó típusát, majd visszatér a BASIC-programba.

1710—1910: A veremmutató csökkentésének rutinja.

1710—1760: Csökkenti a veremmutatót.

1770—1830: Ellenőrzi, hogy túlléptük-e a KERNAL ROM alját. Ha igen, akkor az UGRASLE rutin a BASIC ROM tetejére állítja a veremmutatót.

1840—1910: Ellenőrzi, hogy nincs-e tele a verem. Ha igen, akkor kiírja a „STACK FULL" hibaüzenetet.

1920—1960: Az UGRASLE szubrutin.

2000— POP-funkciót megvalósító programrész.

2000—2010: Ellenőrzi az inicializálást, betölti a változó értékének címét, és ellenőrzi, hogy szöveg-e.

```
(c)1983 profi-ass 64
1000:                .opt p,oo
                   ;
                   ;
                   ; veremkezelo rutin
                   ;
                   ;
1060: adress     =   $b08b
1070: chrget     =   $73
1080: error      =   $a445
                   ;
1100: vmut       =   $fe
1110: cim        =   $47
1120: strflag    =   $0d
1130: intflag    =   $0e
                   ;
1150:            *=  $c000
                   ;
1170:            jmp  init
1180:            jmp  push
1190:            jmp  pop
                   ;
1210: el         nop
1220:            nop
                   ;
1240: init       lda  #$ff
1250:            sta  $fd
1260:            sta  $fe
1270:            sta  $ff
1280:            rts
                   ;
                   ; push
                   ;
1320: push       lda  $fd
1330:            cmp  #$ff
1340:            beq  ok1
1350:            jmp  err5
1360: ok1        jsr  elrak
1370:            jsr  chrget
1380:            jsr  adress
1390:            lda  #$ff
1400:            cmp  strflag
1410:            bne  loop1
1420:            jmp  err1
1430: loop1      jsr  intell
1440:            ldx  #$00
1450:            ldy  #$00
1460: bemas      sei
1470:            lda  #$53
1480:            sta  $01
1490:            lda  (cim),y
1500:            sta  (vmut),y
1510:            lda  #$55
1520:            sta  $01
1530:            cli
1540:            inc  cim
1550:            bne  loop2
1560:            inc  cim+1
1570: loop2      jsr  vmcsokk
1580:            inx
1590:            cpx  #$fc
1600:            bne  bemas
1610:            sei
1620:            lda  #$53
1630:            sta  $01
1640:            lda  intflag
1650:            sta  (vmut),y
1660:            lda  #$55
1670:            sta  $01
1680:            cli
1690:            jsr  vmcsokk
1700:            rts
1710: vmcsokk    lda  vmut
1720:            sec
1730:            sbc  #$01
1740:            sta  vmut
1750:            bcs  loop3
1760:            dec  vmut+1
1770: loop3      lda  vmut
1780:            cmp  #$ff
1790:            bne  loop4
1800:            lda  vmut+1
1810:            cmp  #$df
1820:            bne  loop4
1830:            jsr  ugrasle
1840: loop4      lda  vmut
1850:            cmp  #$00
1860:            bne  loop5
1870:            lda  vmut+1
1880:            cmp  #$a0
1890:            bne  loop5
1900:            jmp  err2
1910: loop5      rts
1920: ugrasle    lda  #$ff
1930:            ldy  #$bf
1940:            sta  vmut
1950:            sty  vmut+1
1960:            rts
                   ;
                   ; pop
                   ;
2000: pop        lda  $fd
2010:            cmp  #$ff
2020:            beq  ok2
2030:            jmp  err5
2040: ok2        jsr  elrak
2050:            jsr  chrget
2060:            jsr  adress
2070:            lda  #$ff
2080:            cmp  strflag
2090:            bne  loop6
2100:            jmp  err3
2110: loop6      jsr  vmmov
2120:            ldy  #$00
2130:            lda  #$53
2140:            sei
```

```
2150:           sta  #01
2160:           lda  (vmut),y
2170:           tay
2180:           lda  #55
2190:           sta  #01
2200:           cli
2210:           tya
2220:           cmp  intflag
2230:           beq  loop7
2240:           jsr  vmcsokk
2250:           jmp  err3
2260:  loop7    jsr  intell
2270:           jsr  vmnov
2280:           clc
2290:           lda  cim
2300:           adc  #fb
2310:           sta  cim
2320:           bcc  loop8
2330:           inc  cim+1
2340:  loop8    ldx  ##00
2350:  Kiir     ldy  ##00
2360:           lda  #53
2370:           sei
2380:           sta  #01
2390:           lda  (vmut),y
2400:           sta  (cim),y
2410:           lda  #55
2420:           sta  #01
2430:           cli
2440:           lda  cim
2450:           sec
2460:           sbc  #01
2470:           sta  cim
2480:           bcs  loop9
2490:           dec  cim+1
2500:  loop9    inx
2510:           cpx  #fc
2520:           bne  loop10
2530:           rts
2540:  loop10   jsr  vmnov
2550:           jmp  Kiir
2560:  vmnov    inc  vmut
2570:           bne  loop11
2580:           inc  vmut+1
2590:  loop11   lda  vmut
2600:           cmp  ##00
2610:           bne  loop12
2620:           lda  vmut+1
2630:           cmp  ##c0
2640:           bne  loop12
2650:           jsr  ugrasfel
2660:  loop12   lda  vmut
2670:           cmp  ##00
2680:           bne  loop13
2690:           lda  vmut+1
2700:           cmp  ##00
2710:           bne  loop13
2720:           jmp  err4
2730:  loop13   rts
2740:  ugrasfel lda  ##00
2750:           ldy  ##e0
2760:           sta  vmut
2770:           sty  vmut+1
2780:           rts
2790:  err1     lda  #< erc1
2800:           sta  #22
2810:           lda  #> erc1
2820:           jmp  error
2830:  err2     jsr  vrak
2840:           lda  #< erc2
2850:           sta  #22
2860:           lda  #> erc2
2870:           jmp  error
2880:  err3     lda  #< erc3
2890:           sta  #22
2900:           lda  #> erc3
2910:           jmp  error
2920:  err4     jsr  vrak
2930:           lda  #< erc4
2940:           sta  #22
2950:           lda  #> erc4
2960:           jmp  error
2970:  err5     lda  #< erc5
2980:           sta  #22
2990:           lda  #> erc5
3000:           jmp  error
3010:  elrak    lda  vmut
3020:           sta  el
3030:           lda  vmut+1
3040:           sta  el+1
3050:           rts
3060:  vrak     lda  el
3070:           sta  vmut
3080:           lda  el+1
3090:           sta  vmut+1
3100:           rts
3110:  intell   lda  intflag
3120:           cmp  ##00
3130:           bne  loop14
3140:           lda  ##02
3150:           ldy  ##01
3160:           jmp  loop15
3170:  loop14   lda  ##05
3180:           ldy  ##04
3190:  loop15   sta  #fc
3200:           sty  #fb
3210:           rts
3220:  erc1     .asc "string not examplE"
3230:  erc2     .asc "stack fuL"
3240:  erc3     .asc "type mismatcH"
3250:  erc4     .asc "stack emptY"
3260:  erc5     .asc "no initialiseD"
```

2110—2250: Betölti a veremből a típust, ellenőrzi, hogy megegyezik-e a változó típusával. Ha nem, akkor a megfelelő hibarutinra ugrik ("TYPE MISMATCH").

2260—2330: Meghívja az INTELL rutint, majd $fb értékét hozzáadja a változó címéhez. Erre azért van szükség, mert a változót a verembe alulról felfelé írtuk be, és mivel a veremből mindent fordítva kapunk vissza, felülről lefelé kell visszamásolni.

2340—2550: A másoló ciklus. A PUSH rutinnál leírthoz hasonlóan működik, csak a veremmutatót növeljük, a változó címét pedig csökkentjük.

2560—2730: A veremmutató értékét növelő szubrutin. Ellenőrzi a BASIC felső határt és a verem kiürülését.

2740—2780: Ha elértük a BASIC felső határt, akkor a KERNAL aljára állítja a veremmutatót.

2790—3000: A hibaüzenetek kiírása.

3010—3050: Az ELRAK szubrutin. A veremmutató értékét a rutinok végrehajtásának idejére az EL címkéjű tárcímre másolja, így hiba esetén a veremtartalom nem vész el.

3060—3100: A VRAK szubrutin. Hiba esetén visszamásolja az EL címkéről a veremmutatót.

3110—3210: Az INTELL szubrutin. Beállítja a másolandó bájtok számát és egy segédváltozót a visszamásolás kezdőcímének meghatározásához. Ha a változó egész, akkor a $FC címre kettőt, a $FB címre egyet, egyébként ötöt és négyet tesz. A $FC cím a ciklusváltozót ellenőrzi, a $FB pedig a POP rutinnál a változó címének növeléséhez kell.

3220—3260: A hibaüzenetek szövege. A szöveg utolsó betűjét SHIFT-tel vigyük be!

GNÄDIG PÉTER

Adatbázi

1969-ben jelentette meg a CODASYL rendszerbizottság híres kiadványát, a „Feature Analysis"-t (képességelemzés). Tucatnyi szakember írta le az adatbankrendszerek kívánatos és a létező kezelők tényleges képességeit. A mű a kiadás pillanatára elavult. Új adatbázis-elméletek születtek, és az elméleteken túlmutató gyakorlati kezelők kerültek piacra.

Ha a rugalmatlanabb nagygépek esetén ilyen volt a helyzet, akkor merjek-e szólni a mikrogépes adatbázis-kezelésről? Amit mondok, esetleg már tegnap sem volt igaz! Mivel azonban a téma sokakat érdekel, vállalkozom egyfajta összegzésre. Azzal a kéréssel, hogy az olvasó értse meg egyszemélyes „bizottságom" elvi és gyakorlati korlátait, amelyek annak ellenére léteznek, hogy mintegy 5—600 cikket, kiadványt, tájékoztatót, alkalmazási beszámolót, uram bocsá': reklámot olvastam el ebben a témában.

Lesz szó képességekről, hatékonyságról, nyelvekről és kezelési módokról, fizikai és logikai adatszerkezetről, ahogy illik. Egyelőre azonban a legáltalánosabb elveket tekintem át. Létezhetnek-e, vannak-e adatbázis-kezelők a mikroszámítógépeken? Ezt a kérdést az adatbázisok legfontosabb ismérvei szerint vizsgálom meg.

Összetett adatszerkezetek

Az adatbázisok abban különböznek az egyéb módon szervezett adatállományoktól, hogy bennük a különböző jelenségeket tükröző adatok között is szoros, adatokon alapuló kapcsolat van. Ez az összefüggés megteremthető összetett *fizikai* adatszerkezettel (indexek, láncok, mutatók segítségével), de létrehozható összetett *logikai* adatszerkezettel is úgy, hogy az összefüggéseket közös adattételek valósítják meg. Az összetett szerkezetek egyik motívuma az adatok felesleges fizikai és logikai átfedésének (redundanciájának) kiküszöbölése, miközben gondosan ügyelünk a szükséges átfedések megteremtésére.

A kiadványokban, melyeket átnéztem, az adatbázisként megjelölt termék az esetek 85—90%-ában(!) egyetlen adatállomány igen korlátozott kezelésére volt képes. Egyes eladók nem átallottak egyszerű lekérdezőt, táblázatkezelőt, sőt rendezőrutint adatbázis-kezelőnek nevezni!

Találkoztam viszont biztató jelekkel is. Itt van — a hálós adatbázisokra példaként — az MDBS (mikro adatbázis-kezelő rendszer), amely a CODASYL-javaslatokon is túlmenő szerkezetek (M:N-es viszonyok, visszamutatások) kezelésére képes. A témát kedvelők közül pedig ki ne hallott volna a dBASE III kezelőről, amit relációsnak titu-

ezelés

Általános áttekintés

lálnak? Hasonlít a relációsra, de nem az. A következőkben ezt látni fogjuk. De ettől nem kell megijedni, igen prima kis szerszám az a dBASE III! Több adatállomány kapcsolt kezelésére alkalmas, és ezzel kimeríti az adatbázisok első ismérvét. Vagyis: léteznek, nemcsak létezhetnek mikrogépes adatbázis-kezelők.

Még várni kell az igaziakra. A dBASE III és relációs követői (rBASE, Knowledgeman stb.) eljárási úton teremtik meg a logikai összefüggéseket. Ha azt mondom: SET RELATION TO . . ., akkor van kapcsolat. Ha nem mondom, akkor nincs. Vagyis ez a képesség csak félutat jelent a hagyományos MERGE (összeválogató) rutintól a logikai szintű adatkezelés felé. Félutat, de jó utat.

Adatfüggetlenség

Az adat—program-függetlenség elve azt jelenti, hogy az adatszerkezetben, tágabb értelemben pedig a program környezetében bekövetkező változások nem okozzák szükség szerint a program módosítási igényét. Ez az ismérv igen összetett. Beszélhetünk logikai adatfüggetlenségről, amikor a logikai *adatszerkezetben* lévő változás hagyja változatlanul a programot. Ismeretes a fizikai adatfüggetlenség, amikor a tárolási-reprezentálási *adatszervezésben* végrehajtott módosítás hagyja hidegen a programot. De milyen függetlenségnek nevezzük azt az elvet, hogy például ne soroljuk fel az ellenőrzendő kódokat a programban, mert holnap bővülhet készletük, programváltozást okozva?

A függetlenségnek nincs gyakorlati mércéje. Elméleti már van. Az ISO ún. „100%-os elv"-e kimondja: a program csak eljárást tartalmazzon. Az adatokra, a program működési környezetére, végrehajtási feltételeire stb. vonatkozó környezeti információk száz százalékig a programtól független leírásokban kell hogy megtestesüljenek. Tájékoztatásul megemlítem, hogy egy hagyományos COBOL programnál a függetlenség mértéke mintegy 10%-os. A CODASYL típusú, nagygépes adatkezelőknél alig haladja meg a 20%-ot.

Bátor vállalkozás lenne becslésekbe bocsátkoznom. Megkockáztatom azonban, hogy ezt a 20%-os mércét a mikrogépes rendszerek is teljesítik. Az MDBS-ben a teljes CODASYL sémarendszer érvényesül. A dBASE III-ban viszont a tárváltozók kijelölésénél, a kommunikációban elég sok az ábrázolási megkötés. Az MDBS fizikai mutatókkal operál, vagyis fizikai függő. A dBASE logikai kapcsolatokra épít, látszólag tehát független. Vagyis: az adatbázisok második fontos kritériuma alapján is azt kell mondanom, hogy nemcsak létezhetnek, de valóban léteznek is mikrogépes adatbázis-kezelők.

No, nem tökéletesek. Mert — például az SQL/DS-szel, a relációs nagygépi IBM rendszerrel szemben — dBASE III-ban azt kell mondanom, hogy SET INDEX TO . . ., ha index alapján akarok keresni, és nevetségesen fontos az állományok megnyitási sorrendje. Kiskorúságból fakadó hibák, kényelmetlenségek. Igen biztató kezdet után kitűnő jövő áll a mikrogépes adatbázisok előtt.

Osztott hozzáférés

Nem tévesztendő össze a fizikai távolságra utaló elosztott (distributed) kifejezéssel. Elosztott adatbázis még nagygépeken sincs. Ha valaki mást állít, az nem ért a témához.

Az osztott hozzáférés azt jelenti, hogy ugyanazt az adatdarabkát több felhasználó (látszólag) egyidejűleg is kezelni tudja. Azért csak látszólag, mert a tényleges kezelés mindig adott sorrendű, de gyorsasága miatt czt a felhasználó nem érzékeli. Nos, ezen a téren van a legtöbb lemaradás a mikrók esetében. Nem értek az alapszoftverekhez, ezért lehet, hogy csacsiságot mondok. Én úgy érzékelem, hogy e szoftverek hierarchiája, egymás közötti kommunikációjuk megoldási módja, mindehhez pedig a tárbeli elrendezés megfelelő támogatása még hiányzik, illetve tökéletlen a nagygépekkel való összehasonlításban.

Ez nem véletlen, elvégre eredetileg személyi használatú gépekről volt szó. Szorosan ide kapcsolódik a következő ismérv is, de annak elemzése előtt néhány példát említek. Az MDBS többfelhasználós. Létezik már a dBASE III-nak is ilyen változata. De milyen áron! Ha már ketten használjuk a dBASE-t, akkor a feldolgozási idő háromszorosára (egy konkrét mérés szerint 293%-ára) nő.

Mindez azt mutatja, hogy harmadik ismérvünk terén nagyobb és elméletileg is jelentősebb a mikrogépes adatbázisok lemaradása. De beszélünk az osztott felhasználás lehetőségeiről. Vannak első és igéretes próbálkozások. Vagyis: ebből a szempontból is azt kell mondanom, hogy létezhetnek, és — az elmélet dogmájának szelídítésével értékelve — léteznek is adatbázis-kezelők a mikrókon. Fenntartásomat jelzi a következő kérdéskör.

Általánosított kezelőrendszer

Egyelőre úgy kell fogalmaznom, hogy számos olyan mikrogépes rendszer létezik, amely „mímeli" az adatbázis-kezelést. Az „igazi" adatbázis-kezelő nemcsak nyelv, nemcsak rutinok gyűjteménye. A valódi adatbázis-kezelőknek van egy „adatbázis-vezérlő" magjuk, amit tekinthetünk logikai szintű operációs rendszernek is. E mag feladata az ütemezés, a kizárás, a helyreállítás, a sorbaállítás, a sémakezelés, a rutinhívások összhangjának megteremtése stb.

Sajnálatos tény, hogy az egyáltalán nem mellékes biztonsági, helyreállítási, hozzáférés-védelmi funkciók terén a mikrogépes adatkezelők igen elmaradottak. Ez részben már a konfiguráció problémáiból is következik. Vagy lenne-e bárkinek kedve kimenteni egy mindössze 20 Mbájtos rögzített lemezt diszkettekre?

Sokan úgy vélik, hogy a mikrogépes adatkezelők 500—2000 dolláros ára pusztán reklámár. Tévedés! Ezek a rendszerek ennyit tudnak. Az általánosított kezelőrendszer funkcióival jól ellátott MDBS ára meghaladja a 30 ezer dollárt. Nem véletlenül. Elvégre egy operációs rendszer szintű termék fejlesztése nem bagóba kerül.

Ugyanakkor az MDBS-példa is mutatja, hogy létezhetnek mikrókon adatbázis-kezelő *rendszerek* is, nemcsak korlátos adatbázis-kezelő *funkciók*. Remélem, az olvasó érti a különbséget.

Egy félreértésről

Hallottam már olyan kitételt, hogy a mikrókon nem lehetnek adatbázisok, mert ahhoz túlzottan kicsik és lassúak. Nos, a mikrók nem kicsik. A nagy IBM-en 750 kbájtos központi egység van, az előttem lévő IBM PC AT-n lehet akár 3 Mbájt is. Pár éve még összetettük volna a kezünket annyi háttértár láttán, ami egy AT-re kapcsolható. A félreértés azonban még ennél is súlyosabb.

Az adatbázis nem attól az, mert nagy. Új fogalmat akkor alkotunk, ha új minőséggel nézünk szembe. Az állománykezelést az általam felsorolt, de egyáltalán nem általam kitalált, fenti kritériumok alapján kell megítélni. Vagyis: igenis létezhetnek adatbázisok mikrókon. Vannak is. De erről később.

* * *

A folytatásban sorra fogom venni az adatkezelők legfontosabb kritériumait, példákkal illusztrálva a mondanivalót. Aki némi csalódást érez az eddigi véleménnyel kapcsolatban, az elfelejti, hogy egy oldalon állunk. Szeretnénk minél jobb, minél szolgálatkészebb adatkezelőt kis barátunkon, a mikrogépen. Ha tehát maximalista vagyok, akkor a jövőre is nézek, nemcsak a sokakat máris boldogító jelenre.

Kérem az olvasókat, hogy bátran tegyék fel elvi és konkrét kérdéseiket. Ha tudok, válaszolok. Örülnék, ha párbeszéd lenne a monológból.

DR. HALASSY BÉLA

COMMODORE 64

Tömörítő rutin

A C—64-hez használt floppymeghajtók kapacitása elég szűkös, ennek ellenére a lemezen a numerikus adatok tárolása karakteres formában történik. Ezért készítettem egy assembly rutint *(1. lista)*, amely a numerikus sztringet pakkolt formába tömöríti. Ez minden további nélkül felírható a lemezre. A *2. listán* látható rutin a pakkolt formátumot visszaalakítja karakteressé. E két rutin segítségével 30—45 százalékkal lehet növelni egy lemez kapacitását, a kiírandó numerikus adatok átlagos hosszától függően.

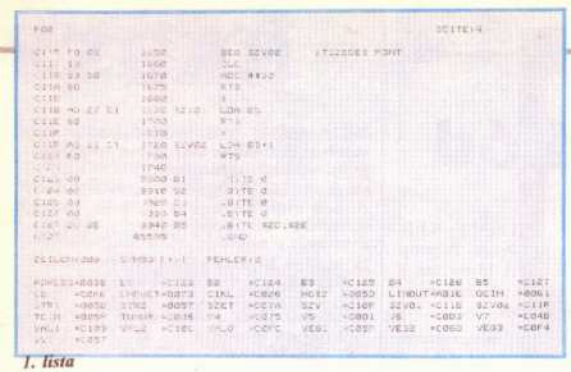
Összehasonlításképpen 1-től 1000-ig kiírattam a lemezre az összes számot, a rutinok segítségével tömörített formában és hagyományos módon is. A alábbi adatokat mértem:

	Hagyományos	Rutin segítségével	
Írás a lemezre [ms]:	1013	1137	112%
Olvasás a lemezről [ms]:	1048	914	87%
A fájl mérete blokkban:	272	193	70%

Ha valamilyen adatot nem közöltem, vagy a forrásprogram nem elég áttekinthető, további felvilágosítást, magyarázatot is szívesen adok.

MAROFKA FERENC

```
1000           100         ;
1000           110         ; EGY KARAKTERES VALTOZO TOMORITESE
1000           120         ;
1000           130         ;      ZONAZOTT ALAKRA FORMALAS
1000           140         ;
1000           150         ; ********************************************
1000           160         ;
1000           170         ;       HIVASA = SYS 50000,A$
1000           180         ;
1000           190         ; AHOL A$ - EGY NUMERIKUS ADAT
1000           191         ;
1000           192         ;  FIGYELEM ! AZ ELSO HIVAS ELOTT AZ A$-NEK
1000           193         ;             ADJUNK KEZDO ERTEKET,AMI EGY
1000           194         ;             LEGALABB 10 HOSSZU STRING LEGYEN
1000           200         ;
1000           210         ; ********************************************
1000           220         ;
B08B           230 ADRESS =$B08B    ; RENDSZER RUTINOK
0073           234 CHRGET =$73      ;
AB1E           240 LINOUT =$AB1E    ;
005F           250 TCIM   =$5F      ; A VALTOZO LEIRO BLOKK KEZDO CIME
0058           260 STR1   =$58      ; A VALTOZO KEZDOCIME
0050           270 HOSZ   =$50      ; A VALTOZO HOSSZA
0061           275 OCIM   =$61
0057           280 STR2   =$57
1000           290         ;
C000           300         *=49152
C000           310         ;
C000 4C 06 C0  312         JMP TOMOR   ; TOMORITES
C003 4C 7A C0  314         JMP SZET    ; SZETHUZAS
C006 20 73 00  320 TOMOR   JSR CHRGET  ; A TERMINATOR BEOLVASASA
C009 20 8B B0  330         JSR ADRESS  ; A CIM KERESES
C00C A0 02     340         LDY #2
C00E A9 00     341         LDA #0
C010 8D 24 C1  342         STA B2
C013 B1 5F     350         LDA (TCIM),Y ;VALTOZO LEIRO TOMBBOL A HOSSZ OLVASASA
C015 85 50     360         STA HOSZ     ; A VALTOZO HOSSZMENTES
C017 C8        370         INY
C018 B1 5F     380         LDA (TCIM),Y ; AZ ADAT KEZDOCIMENEK A MENTESE
C01A 85 58     390         STA STR1
C01C C8        400         INY
C01D B1 5F     410         LDA (TCIM),Y
C01F 85 59     420         STA STR1+1
C021           430         ;
C021 A0 00     440         LDY #0
C023 8C 26 C1  445         STY B4         ; A BYTE SZAMLALO NULLAZASA
C026 B1 58     450 CIKL    LDA (STR1),Y   ; A VALTOZO KOVETKEZO BYTEJENEK A BEOL
VASASA
C028 20 FC C0  460         JSR VALO
C02B 18        465         CLC
C02C 0A        470         ASL A          ; LEPTETES A FELSO TETRADRA
C02D 0A        480         ASL A
C02E 0A        490         ASL A
C02F 0A        500         ASL A
C030 8D 23 C1  510         STA B1         ; MENTES
C033 C8        520         INY
C034 C4 50     525         CPY HOSZ       ; VAN-E MEG VIZSGALANDO BYTE
C036 F0 27     530         BEQ VEG1       ; NINCS VEGE A FELDOLGOZASNAK
C038 B1 58     540         LDA (STR1),Y   ; A KOVETKEZO BYTE BEOLVASASA
C03A 20 FC C0  550         JSR VALO
C03D 18        555         CLC
C03E 6D 23 C1  560         ADC B1
C041 8C 25 C1  564         STY B3
C044 AC 26 C1  566         LDY B4
C047 C9 00     567         CMP #$00
C049 F0 2A     568         BEQ V4
C04B 91 58     570 V7      STA (STR1),Y
C04D C8        580         INY
C04E 8C 26 C1  584         STY B4
C051 AC 25 C1  590         LDY B3
C054 EE 24 C1  600         INC B2
C057 C8        610 VV1     INY
C058 C4 50     640         CPY HOSZ
C05A F0 11     650         BEQ VEG2   ; NINCS TOBB VIZSGALANDO BYTE
C05C 4C 26 C0  660         JMP CIKL
C05F A9 FF     670 VEG1    LDA #15    ; CSAK A FELSO BYTE-RA JUTOTT TETRAD MIVEL
C061 18        675         CLC                 ; PARATLAN HOSSZUSAGU AZ INPUT STRING
C062 6D 23 C1  680         ADC B1
C065 AC 26 C1  685         LDY B4
C068 91 58     690         STA (STR1),Y
C06A EE 24 C1  692         INC B2
C06D           700         ;
C06D AD 24 C1  710 VEG2    LDA B2
C070 A0 02     720         LDY #2
C072 91 5F     730         STA (TCIM),Y
C074 60        740         RTS
C075           741         ;
C075 A9 0C     742 V4      LDA #12
C077 4C 4B C0  744         JMP V7
C07A           750         ;
C07A           760         ;
C07A           770         ;    SZETHUZAS - PAKOLTBOL - DECIMALIS
C07A           775         ;    HIVASA SYS 49155,B$,A$
C07A           776         ;    AHOL B$ = AZ EREDMENY VALTOZO
C07A           777         ;    AHOL A$ = AZ INPUT VALTOZO
C07A           780         ;
C07A           790         ;
C07A 20 73 00  800 SZET    JSR CHRGET    ; A VESSZO
C07D 20 8B B0  810         JSR ADRESS         ; A CIM KERES
C080 A0 03     820         LDY #3
C082 B1 5F     830         LDA (TCIM),Y       ; SZOVEG KEZDO CIME
C084 85 58     840         STA STR1
C086 C8        850         INY
C087 B1 5F     860         LDA (TCIM),Y
C089 85 59     870         STA STR1+1
C08B A5 5F     880         LDA TCIM           ; AZ OUTPUTCIM MENTESE
C08D 85 61     890         STA OCIM
C08F A5 60     900         LDA TCIM+1
C091 85 62     910         STA OCIM+1
C093           920         ;
C093 20 73 00  930         JSR CHRGET         ; VESSZO OLVASAS
C096 20 8B B0  940         JSR ADRESS         ; A CIM KERESES
C099 A0 02     950         LDY #2
C09B B1 5F     960         LDA (TCIM),Y       ; AZ INPUT HOSSZA
C09D 85 50     970         STA HOSZ           ; TOMORITETT
C09F           980         ;
C09F C8        990         INY
C0A0 B1 5F     1000        LDA (TCIM),Y     ; KEZDO CIM MENTESE
C0A2 85 57     1010        STA STR2
C0A4 C8        1020        INY
C0A5 B1 5F     1030        LDA (TCIM),Y
C0A7 85 58     1040        STA STR2+1
C0A9           1050        ;
C0A9 A0 00     1060        LDY #0
C0AB 8C 26 C1  1070        STY B4
C0AE           1080        ;
C0AE B1 57     1090 CE     LDA (STR2),Y   ; BYTE OLVASA
C0B0 29 F0     1100        AND #$F0
C0B2 18        1110        CLC
C0B3 6A        1120        ROR A
C0B4 6A        1130        ROR A
C0B5 6A        1140        ROR A
C0B6 6A        1150        ROR A
C0B7 20 0F C1  1160        JSR SZV
C0BA 8C 23 C1  1170        STY B1
C0BD AC 26 C1  1180        LDY B4
C0C0 91 58     1190        STA (STR1),Y
C0C2 EE 26 C1  1200        INC B4
C0C5           1210        ;
C0C5 AC 23 C1  1220        LDY B1
C0C8 B1 57     1230        LDA (STR2),Y
C0CA C9 0C     1232        CMP #12
C0CC F0 03     1234        BEQ V5
C0CE 4C D3 C0  1236        JMP V6
C0D1 A9 00     1238 V5     LDA #0
C0D3 29 0F     1240 V6     AND #$0F
C0D5 18        1245        CLC
C0D6 C9 0F     1250        CMP #$0F
C0D8 F0 1A     1260        BEQ VEG3
C0DA 18        1270        CLC
C0DB 20 0F C1  1280        JSR SZV
C0DE 8C 23 C1  1290        STY B1
C0E1 AC 26 C1  1300        LDY B4
C0E4 91 58     1310        STA (STR1),Y
C0E6 EE 26 C1  1320        INC B4
C0E9           1330        ;
C0E9 AC 23 C1  1340        LDY B1
C0EC C8        1345        INY
C0ED C4 50     1350        CPY HOSZ
C0EF F0 03     1360        BEQ VEG3
C0F1 4C AE C0  1380        JMP CE
C0F4           1390        ;
C0F4 AD 26 C1  1400 VEG3   LDA B4     ; A HOSZ MENTESE
C0F7 A0 02     1410        LDY #2
C0F9 91 61     1420        STA (OCIM),Y
C0FB 60        1430        RTS
C0FC           1440        ;
C0FC           1450        ;   VALOGATAS
C0FC           1451        ; A MINUSZ JEL ES A TIZEDESPONT VIZSGALATA
C0FC           1452        ;  MIVEL EZ IS JELOLVE VAN A PAKOLT SZAMBAN
C0FC           1453        ;        HEX 10 = MINUSZ JEL
C0FC           1454        ;        HEX 11 = TIZEDES PONT
C0FC           1455        ;
C0FC           1460        ;
C0FC CD 27 C1  1470 VALO   CMP B5   ; MINUSZ
C0FF F0 08     1475        BEQ VAL1
C101 CD 28 C1  1480        CMP B5+1
C104 F0 06     1490        BEQ VAL2      ; TIZEDES PONT
C106 29 0F     1500        AND #$0F
C108 60        1510        RTS
C109           1520        ;
C109 A9 0A     1530 VAL1   LDA #10
C10B 60        1540        RTS
C10C           1550        ;
C10C A9 0B     1560 VAL2   LDA #11
C10E 60        1570        RTS
C10F           1580        ;
C10F           1590        ;
C10F           1600        ; A PAKOLT SZAM ELLENORZESE
C10F           1610        ;
C10F C9 0A     1620 SZV    CMP #10   ; MINUSZ
C111 F0 08     1630        BEQ SZVB1
C113 C9 0B     1640        CMP #11
```



1. lista

2. lista

VC 20 FOLTTERVEZŐ

Aki nem szeret bináris számokra átváltani, használhatja az alábbi programot, amellyel karaktereket lehet tervezni.

A program működése

10—90 a képernyő előkészítése, grafikabetöltés

100—210 változók beállítása, képernyőgrafika

220—350 főprogram
(működése egyszerű, logikus, könnyen érthető az értelmezésben segitenek a változók:)
A — a villogó karakter helyét számítja, tartalmazza
X — a vizszintes mozgás változója
E — az adott cím értékét tartalmazza
D tömb — az adott cím értékét tárolja
T — az adatkiírás helye
C — a cimet számolja (7168—7175)

400—420 elraktározza a „nagyitott" karaktert

430—460 mozgatja a karaktert

470—500 kimásolja a karaktert, visszatér a főprogramba.

A kurzort jelképező négyzet villogását lassithatjuk, ha a főprogramba REM sorokat irunk, amelyekbe grafikai jeleket töltünk.

A 240-es és 300-as sorokat röviditve kell begépelni.

A 130-as sorban RVS ON után C=billentyű+N, RVS OFF, 8 db A betű, RVS ON, C=billentyű+H áll.

HAJNAL CSABA

```
 1 REM ********************************************
 2 REM *                                          *
 3 REM * NUMERIKUS STRING ES PAKOLT SZAM KONVERZIO *
 4 REM *                                          *
 5 REM * BASIC BETOLTO PROGRAM                    *
 6 REM * HIVASOK : SYS 49152,A$ - AHOL -          *
 7 REM *                   A$ = INPUT,OUTPUT      *
 8 REM *             SYS 49155,B$,A$ AHOL         *
 9 REM *                   A$ = INPUT             *
10 REM*                    B$ = OUTPUT            *
11 REM*                                           *
12 REM*   MEZOTUR 1986.08     ---   MAROFKA FERENC *
13 REM*                                           *
14 REM********************************************
15 :
19 A=0:FOR I=49152 TO 49452
20 READ B:POKE I,B:A=A+B:NEXT I
30 IF A<>34927 THEN PRINT "DATA HIBA !!":STOP
100 DATA  76,  6,192, 76,122,192, 32
110 DATA 115,  0, 32,139,176,160,  2
120 DATA 169,  0,141, 36,193,177, 95
130 DATA 133, 93,200,177, 95,133, 91
140 DATA 200,177, 95,133, 92,160,  0
150 DATA 140, 38,193,177, 91, 32,252
160 DATA 192, 24, 10, 10, 10, 10,141
170 DATA  35,193,200,196, 93,240, 39
180 DATA 177, 91, 32,252,192, 24,109
190 DATA  35,193,140, 37,193,172, 38
200 DATA 193,201,  0,240, 42,145, 91
210 DATA 200,140, 38,193,172, 37,193
220 DATA 238, 36,193,200,196, 93,240
230 DATA  17, 76, 38,192,169, 15, 24
240 DATA 109, 35,193,172, 38,193,145
250 DATA  91,238, 36,193,173, 36,193
260 DATA 160,  2,145, 95, 96,169, 12
270 DATA  76, 75,192, 32,115,  0, 32
280 DATA 139,176,160,  3,177, 95,133
290 DATA  91,200,177, 95,133, 92,165
300 DATA  95,133, 97,165, 96,133, 98
310 DATA  32,115,  0, 32,139,176,160
320 DATA   2,177, 95,133, 93,200,177
330 DATA  95,133, 87,200,177, 95,133
340 DATA  88,160,  0,140, 38,193,177
350 DATA  67, 41,240, 24,106,106,106
360 DATA 106, 32, 15,193,140, 35,193
370 DATA 172, 38,193,145, 91,238, 38
380 DATA 193,172, 35,193,177, 87,201
390 DATA  12,240,  3, 76,211,192,169
400 DATA   0, 41, 15, 24,201, 15,240
410 DATA  26, 24, 32, 15,193,140, 35
420 DATA 193,172, 38,193,145, 91,238
430 DATA  38,193,172, 35,193,200,196
440 DATA  93,240,  3, 76,174,192,173
450 DATA  38,193,160,  2,145, 97, 96
460 DATA 205, 39,193,240,  8,205, 40
470 DATA 193,240,  6, 41, 15, 96,169
480 DATA  10, 96,169, 11, 96,201, 10
490 DATA 240,  8,201, 11,240,  8, 24
500 DATA 105, 48, 96,173, 39,193, 96
510 DATA 173, 40,193, 96,  0,  0,  0
520 DATA   0, 45, 46,255,255,255,255
READY.
```

```
0  rem *Folt-tervezo*
10 Print"{clrh}" Poke36869,255 Poke36879
   ,122 Poke55,255 Poke56,23
20fora=7208to7679 POa,0 next
30 fora=7176to7207 readv Pokea,v next
40data0,0,0,0,0,0,24,24,0
50 data255,255,255,255,255,231,231,255
60 data0,60,126,126,126,126,60,0
70 data255,195,129,129,129,129,195,255
80 Poke650,128 Poke657,128 Poke788,194
90fort=7168to7175 Poket,0 next
100 clr dimd(63) c=7168 t=54 e=128
110 Print"{shft+clrh}{rvson}{7*crsr}obb}
    COMMODORE VC20"
120 Print"{crsr}obb}{rvs9*}--------"
130 forr=0to7 Print"{rvson}{rvsoff}
     AAAAAAAA{rvson}";d(r) next
140 Print"{crsr}obb}{rvson}--------"
150 Print"{crsr}le}{5*crsr}obb}{rvson}
    FOLT TERVEZO"
160 Print"{5*crsr}obb}{rvson}--------"
170 Print"E-fel   C-toltes"
180 Print"X-le   V-torles"
190 Print"D-jobbra   B-mozgas"
200 Print"S-balra    N-start"
210 Print"{clrh}"tab(17)"";:ifv=1then
    return
220 Geta$:a=7725+22*y+x
230 ifa$="S"andx>0thenx=x-1 e=e-8
240 ifa$="C"andPeek(A)<>3thenPokea,3:
    d(y)=d(y)+e:Pokec,d(y):Print"{clrh}"
    tab(t)"     "
    Print"{clrh}"tab(t)"{rvson}";d(y)
250 Pokea,Peek(a)+1
260ifa$="D"andx<7thenx=x+1 e=e/2
270 ifa$="B"theny=y+1:gosub400
280 Pokea,Peek(a)-1
290 ifa$="E"andy>0theny=y-1:c=c-1 t=t-22
300 ifa$="V"andPeek(a)<>1thenPokea,1
    :d(y)=d(y)-e Pokec,d(y):Print"{clrh}"
    tab(t)"     "Print"{clrh}"tab(t)"{rv
    son}";d(y)
310 Pokea,Peek(a)+1
320 ifa$="X"andx<7thenx=x+1 c=c+1 t=t+22
330 ifa$="N"then90
340 Pokea,Peek(a)-1
350 goto220
400 g=0
410 fori=7725to7887step22
420 forz=itoi+7 k(g)=Peek(z) g=g+1
    nextz,i
430 Print"{shft+clrh}"tab(225)"{rvson}
    FOLT-TERVEZO"
440 forn=0to2*CPi)stepCPi/90 g=int(11+
    9*cos(n)) w=int(11+8*sin(n)) w=k
    b=7680+22*wb
450 Pokek+38720,0:Pokek,0 ifu<>kthenPoke
    u,160
460 next:forn=0to500:next:v=1:gosub110
    :v=0
470 g=0
480 fori=7725to7887step22
490 forz=itoi+7:Pokez,k(g):g=g+1:nextz,i
500 return
```

BASIC és gépi kód

Legutóbb megismerkedtünk az A regiszter és az indexregiszterek közötti adatátvitelt lebonyolító utasításokkal, néhány címzési móddal, továbbá egy hasznos gépi kódú programmal és annak használatával. Most ennek a programnak a működését nézzük meg, és szó lesz egy véletlen felfedezésről.

A SAVE program működése

Az itt következő leírásban az előző számban közölt assembly (pontosabban disassembly) listák utasításcímeire fogok hivatkozni. A két változat — mint említettem — csak a hivott ROM-rutinok címeiben különbözik.

A végén kezdjük, a $02F7 címen kezdődő szubrutinnal. Ezt a főprogramból kétszer hívjuk meg, a SYS utasítás egy-egy numerikus paraméterét olvassa be a BASIC sorból, és helyezi el a LINNUM rendszerváltozóban.

A $02F7-ről hívott CHKCOM rutinnal már korábban találkoztunk, ez a paramétereket elválasztó vessző jelenlétét vizsgálja.

$02FA-ról az FRMEVL rutint hívjuk. Ez a BASIC programsorban található aritmetikai vagy karakterlánc-kifejezést értékeli ki, a szövegmutató (TXTPTR) által mutatott helytől a kifejezés feltételezett végéig, és közben a mutató értékét is módosítja. A kifejezés típusának megfelelően egy jelzőt helyez el a $0D címen, $00-t, ha a kifejezés numerikus, $FF-t, ha karakterlánc. Aritmetikai kifejezés esetén a kiértékelés utáni végeredmény a lebegőpontos akkumulátorba (FAC) kerül.

A következő utasítás feltétel nélküli vezérlésátadás a GETADR rutinra, melynek végrehajtása után a program futása a JSR $02F7 utáni utasítással folytatódik. A GETADR a FAC-ban talált számot 16 bites előjel nélküli egésszé alakítja, a LINNUM rendszerváltozóba teszi, a $14 . . . $15 címekre.

Lássuk ezután a főprogramot:

$02C8: a CHKCOM-mal ellenőrizzük az első paraméter (a fájlnév) előtti vesszőt.

$02CB: a FRMEVL rutinnal feldolgozzuk a fájlnevet tartalmazó karakterlánc-kifejezést.

$02CE: meghívjuk a karakterlánc-kifejezés eredményének a paramétereit feldolgozó rutint. Ez a karakterlánc kezdőcímét a $22 . . . $23 címeken levő rendszerváltozóba teszi, a hossza pedig az A regiszterbe kerül. Ennél az utasításnál egy kis pontatlanságot követtem el, melynek magyarázatára később még visszatérek.

$02D1 . . . $02D5: a SETNAM KERNAL rutin előkészítése és hívása. A programnév kezdőcímét a LINNUM-ból az X és Y regiszterekbe töltjük, a név hossza az A regiszterben már benne van.

$02D8 . . . $02DD: a fájl paramétereit beállító SETLFS KERNAL rutin előkészítése és hívása. Az X regiszterbe közvetlen címzési módban betöltjük a fizikai egységszámot, az A regiszterbe a fájl logikai száma kerül, az Y-ba a másodlagos cím. Az egyszerűség kedvéért ez utóbbi kettő megegyezik egymással. A másodlagos cím lemezegység használata esetén a csatornaszámot jelenti (2 és 14 között bármi lehetne), mágnesszalagos tároló esetén a megnyitási módot. Itt kötelező a 2 használata, ami kivitelre való megnyitást jelent, szalagvégjel felírásával.

$02E0 . . . $02E9: a kimentendő program kezdőcímének feldolgozása. A szubrutinnal beolvassuk ezt a kezdőcímet a LINNUM-ba, majd onnan a $FB . . . $FC címekre másoljuk.

$02EB . . . $02F2: a szubrutinnal beolvassuk a kimentendő program vége utáni bájt címét. Ezt a LINNUM-ból az X/Y regiszterpárba tesszük, A-ba közvetlen címzési móddal beírjuk annak a bájtpárnak a címét, ahol a SAVE KERNAL rutin a mentendő program kezdőcímét találja.

$02F4: az előkészítés után a SAVE KERNAL rutinra ugrunk, melynek lefutása után visszakerülünk a BASIC rendszerbe. Az itt szereplő KERNAL rutinok használatáról „Az ismeretlen C16" című sorozatnak az 1986/9. számban megjelent részében írtam bővebben.

Egy véletlen felfedezés

Néhány nappal az előző rész kéziratának leadása után C64-essel és mágnesszalagos egységgel dolgoztam. Egy gépi kódú programot figyelmetlenségből az előírásos forma helyett egyszerű LOAD paranccsal töltöttem a gépbe. Nagyon meglepődtem, mert a program az eredeti helyére töltődött. Bár tudtam, hogy a LOAD KERNAL rutin kazettáról betöltő része felismeri a gépi kódú programokat, de BASIC-ből eddig nem próbálkoztam ilyesmivel. VC-20-assal éppen ellenkező tapasztalatom volt, mert ott a BASIC LOAD parancsa még az 1-es másodlagos címet sem hajlandó felismerni. Nem tudom, hogy ez a jelenség általános-e a C64-eseken, vagy csak egyes sorozataira jellemző. A rendelkezésemre álló irodalomban eddig nem olvastam róla.

BARNA LÁSZLÓ

OKOS K

Társlapunk, a Számítástechnika 1984. októberi számában cikk jelent meg Mikroprocesszoros memóriakártya Franciaországban (és Magyarországon?) címmel. A cikk végigkísérte ennek az új, személyi használatú mikroelektronikai eszköznek a történetét a találmánytól a megvalósításig, és utalt a hazai alkalmazás lehetőségeire.

Nézzük meg most kérdés-felelet formájában — az eszköz rövid ismertetése után —, mi történt azóta a világban és nálunk, milyen tervek merültek fel az „okos kártyával" kapcsolatban?

A fejlett iparú, fogyasztói társadalmi szerkezetű országokban egyre nagyobb gondot okoz a készpénz növekvő forgásával együtt járó időrabló és költséges adminisztratív tevékenység. Ehhez járul a pénz előállításának magas költsége, továbbá a hamisítással, lopással és egyéb visszaélésekkel kapcsolatos veszteségek nagysága is. Ezeken a problémákon alapvetően nem segítettek a széles körben elterjedt készpénzkímélő eszközök sem, mint a csekk, a hitelkártya vagy annak korszerűbb változata, a mágnescsíkos hitelkártya.

A gondok súlyosbodásával egy időben indult meg a mikroelektronika forradalmi fejlődése. Az egyre többet tudó mikroprocesszorok és az egyre nagyobb tárolókapacitású memóriák, a RAM-ok, ROM-ok és azok legkorszerűbb változata, az elektromosan törölhető és újraprogramozható, az információt tápfeszültség nélkül megtartó EPROM-ok ára egyre csökkent, méreteik pedig zsugorodtak.

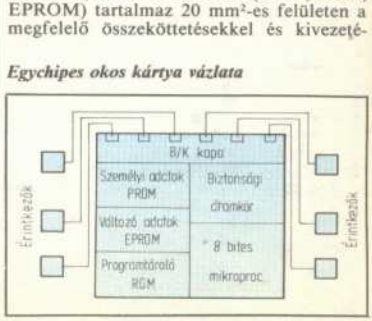
E két körülmény találkozása adta a gondolatot egy francia újságírónak, Roland Morenónak, hogy egy sor szabadalmat nyújtson be a minden eddiginél biztonságosabb hitelkártyára, francia nevén a „carte á mémoire"-ra.

Mi az okos kártya?

A mikroprocesszoros memóriakártya olyan személyes használatú fizetőeszköz, amely a hagyományos hitelkártyák szabványos formátumában készül, és amelybe külső táplálású mikroszámítógép chip van beépítve.

Az okos kártyának, intelligens kártyának vagy aktív memóriakártyának nevezett eszköz 86 mm hosszú, 54 mm széles és 0,76 mm vastag műanyag kártya, amely egy 8 bites mikroprocesszort, be/kimeneti (B/K) áramkört, többféle memóriát (RAM, ROM, EPROM) tartalmaz 20 mm²-es felületen a megfelelő összeköttetésekkel és kivezeté-

Egychipes okos kártya vázlata



RTYÁK

sekkel, az ábrán látható vázlat szerint. Csatlakozója, amelyen a táplálás és adatáramlás történik, 8 aranyozott érintkezőből áll. (Ezekből 6 van kihasználva.)

Hogyan használják?

A kártyát a bankban töltik fel a tulajdonos betétjéből. A feltöltés a bankos és a tulajdonos titkos kódjának együttes alkalmazásával történik.

Vásárlásnál a tulajdonos — miután kártyáját bedugta a pénztárgép külön vevői modulján — bebillentyűzi titkos kódját, a kereskedő a pénztárgép megjelenítőjén ellenőrzi a kártya érvényességét és pénztartalmát (a folyamat automatikus), majd beüti a vásárolni kívánt termék árát. A vevő jóváhagyását modulján a megfelelő gomb lenyomásával jelzi. Ekkor a memóriakártya tartalmából a pénztárgép memóriájába és a kártyába is beíródik az összeg, a dátum, a termék kódja. A kártyában aritmetikai művelet képezi az új egyenleget, a pénztárgép blokkot ad, ezzel a vásárlás megtörtént.

Az ellopott vagy elvesztett kártyát illetéktelen személy nem tudja felhasználni, mert a tulajdonos kódja titkos, és háromszori téves kódbillentyűzés után a kártyába beépített program reteszeli a kimenetet. Ezenkívül a bejelentett ellopott vagy elveszett kártya száma a kiadó bank útján felkerül a pénztárgép tárolójában levő feketelistára, és minden vásárlásnál automatikus ellenőrzés történik.

Az üzlet zárása után a kereskedő titkos kódjának felhasználásával számol el a banknál a napi bevitellel.

Mire való?

Az okos kártya egyszerűbb, nem titkosított változata meghatározott összegtartalommal érme nélküli telefonálásra, metró-, buszjegy helyett és parkoláskor, fizető autópályákon, benzinkutaknál készpénzfizetés helyett használható. Mint előre fizetett kártya, trafikokban, postán, bankfiókokban, szállodákban kapható, és mivel újra nem tölthető, kimerülés után eldobható. Ezek mikroprocesszor és EPROM nélküli, huzalozott logikájú chipekkel ellátott, olcsóbb kártyák.

A fejlett mikroprocesszoros, EPROM-os változat a már ismertetett bolti felhasználáson kívül a fenti egyszerűbb funkciókat is ellátja, azonkívül távfizetésre is felhasználható. Kábeltévés otthoni készülékkel otthonról fizethetők a közüzemi díjak, a televízión megjelenített áruházi katalógusból kiválasztott termék ára, rendelhető színház-, koncert-, vasúti és repülőjegy, amit távfizetés után házhoz szállítanak. Ugyanígy fizethető a videoműsor és az adatbankhozzáférés díja is.

A kártya felhasználható az egészségügyben betegkartonként, krónikus betegségekben (szív-, vese-, cukorbaj) szenvedőknél a betegségre vonatkozó információk tárolójaként, kórházakban pedig a vizsgálati eredményeket tárolhatja.

Az adminisztrációban a papírmunkát csökkentheti: gyári belépőként, a rugalmas munkaidő nyilvántartásra, gépkocsi-nyilvántartásra, egyetemeken leckekönyv helyett, konferenciákon, rendezvényeken akkreditálókártyaként, igazolvány helyett használható.

Műszaki területen mikrogéptároló-bővítőként, műszerdiagnosztikai kártyaként, robotprogramkártyaként történő felhasználása jöhet számításba.

Mindebből látható, hogy a többcélú okos kártya felhasználási lehetősége szinte beláthatatlan.

Az okos kártya vásárláskor készpénz helyett használható

Milyen előnyei vannak?

Az okos kártya megbízható, papír- és munkaidő-takarékos, egyszerűen, gyorsan kezelhető, off-line működésre alkalmas, csak a bank és a postafiók között igényel on-line kapcsolatot, kizárja a hamisítás és a csalás lehetőségét, széles körben felhasználható, viszonylag olcsó, gazdaságos.

Mennyibe kerül?

A mikroprocesszoros EPROM-os okos kártya ára ez év végére 30 francia frankra (kb. 3 dollár) csökken. Ekkorra fut fel az évi többmillió darabszámú tömeggyártás. A kereskedői terminál ára 10—15 ezer frank, bérleti díja pedig 150—200 frank havonta.

Milyen nemzetközi eredmények születtek eddig?

Franciaországban Blois-ban, Caenban és Lyonban 50—50 ezer db kártyával és 200—200 kereskedői terminállal lefolytatott másfél éves sikeres kísérlet után a bankok 1988-ig terjedő időszakra 1,2 milliárd frankot ruháztak be az okos kártya széles körű elterjesztésébe. 1986-tól már havi 300 ezer kártyát gyártanak. Ez év végéig négy nagy francia tartományban vezetik be 2,5 millió kártyatulajdonossal, akik eddig Carte Bleue (francia VISA) és Credit Agricole hitelkártyával rendelkeztek. 1988 végére egész Franciaországban áttérnek az okos kártyára, melyet mágnescsíkkal is ellátnak, hogy a régi pénztárgépeken is használható legyen. Ez 10—12 millió kártyatulajdonost jelent.

Még 1981-ben megalakult a Nemzetközi Memóriakártya Társaság (INTAMIC), amelynek a nyugat-európai és az északamerikai országok a tagjai. A szervezet az egységesítést, szabványosítást tűzte ki céljául. Az USA-ban vállalat alakult a kártyának és terminájának a francia Bull cég licence alapján történő gyártására.

Az amerikai Master-card és American Express hitelkártya-társaság kísérletet folytat az okos kártyára történő áttérésre. Norvégiában, Olaszországban, Spanyolországban szintén kísérlet indult, Bull-vezetéssel. Az NSZK-ban saját okos kártyát gyártanak (Computer-in-Scheckkarte = CIS). A japánok ugyancsak elkezdték az okos kártyák gyártását, méghozzá a japán technikára jellemzően, mindjárt dupla kapacitással.

Mi a helyzet nálunk?

1980-ban kaptuk az első hírt az okos kártyáról. 1981-ben a franciák háromnapos előadás-sorozaton és bemutatón ismertették a magyar szakemberekkel. 1983-ban OMFB-elemzőtanulmány készült a pénzkimélő eszközökről, köztük az okos kártyáról. 1984 októberében megalakult az Aktív Memóriakártya Gazdasági Társaság (AMK GT) 21 tagintézménnyel, az Országos Műszaki Információs Központ és Könyvtár gesztorálásával. A társaság célja az okos kártya hazai elterjesztése, aminek első kísérletei remélhetőleg még az idén elkezdődnek a Skáka Metró Áruházban. (A társaság taglétszáma idén 26-ra bővült.)

Mi várható?

Az okos kártya alkalmazása világszerte gyorsuló ütemben terjed, kapacitása nő: a japánok a 4 kbájtos után a 8, sőt a 16 kbájtos EPROM-ot is megcélozták.

Ha az AMK GT jól dolgozik, ha felsőbb szintről is megkapja a támogatást, és ha sikerül a terminálok kooperációs gyártását beinditani, az okos kártya nálunk is nyerő lehet.

RÓNAI TIBOR

OLVASTUNK...

persze most is, de az alábbiak nem kötődnek közvetlenül olvasmányélményhez. A szerző eredeti elképzelése az volt, hogy egy feladat megoldásának *folyamatába* avatja be az olvasót, de ahogy a cikket írta és ahogy a megoldásban előrehaladva átmeg átírta, rájött, hogy ami a végén megszületett, az inkább a majdnem kész megoldás utólagos magyarázata, bemutatása lett. Úgy járt ő is, mint sokan mások: rájött, hogy az „alkotás" folyamatának bemutatása még egy egyszerű feladatnál sem fér bele egy cikk keretébe. A tapasztalat és a tisztesség mondatja: az „alkotás folyamatát" bemutatni szándékozó jószándékú kísérletek többségének ez a vége: egyenes útként, vagy világos alternatívák közötti választásként mutatják be azt, ami valójában nem volt az. Az ilyen megközelítés amennyire jószándékú, annyira félrevezető is. A szerző tehát bevallja, hogy ami az alábbiakban következik, nem több, mint egy egyszerű, jól definiált feladat, egy lehetséges szakterület (a sakkozás) szempontjából valószínűleg nem is a legjobb, de egy — maximum két — cikk kereteibe éppen beleférő megoldásának bemutatása. Sokak számára ez is tanulságos lehet.

Emlékeztetjük az olvasót: az „ÉLET"-ben a feladat világos megfogalmazása gyakran nehezebb, mint a megoldása. Mi tehát egyszerűsített helyzetből indulunk ki.

A FELADAT:

a számítógép sakkozik a gép kezelőjével — az ellenféllel. Végjátékot játszanak: a gépnek királya és egy bástyája van. Az ellenfélnek csak királya van. A kiinduló állás az alábbi (lásd az 1. ábrát):

a gép (saját) királya f3
bástyája a1
a kezelő (az ellenfél) királya c5
az ellenfél lép először.

Olyan programot kell készíteni, melylyel a gép az ellenfélnek mattot ad.
(A játékot max. 30 lépésre korlátozzuk.)Néhány kérdést a megoldási folyamat elején célszerűnek látszott megválaszolni:

1. Milyen programozási nyelvet válasszunk a probléma megoldásához?
2. Milyen legyen a program vezérlési struktúrája?
3. Hogyan ábrázoljuk az adatokat (a sakktáblát, a „parti" állását, a három figura pozícióját a táblán stb.)?
4. A gép hogyan határozza meg a soron következő lépést?
(Ebben van az egész program lényege.)
5. Milyen legyen az állás ábrázolása a képernyőn, általában milyen legyen az ember—gép kapcsolat?

(1) A szerző az első kérdésre könnyen válaszolt: kedvenc programozási nyelvét — a PASCAL-t — választotta, mert úgy tapasztalta, hogy ezen a nyelven könnyű jól strukturált, jól olvasható programot írni, és mert ez a nyelv a mikrogépet programozók körében is rohamosan terjed.

(2) A program vázát (vezérlési struktúráját) több kísérlet után (melynek ismertetésétől itt most eltekintünk) az alábbiakban határozta meg:

```
(*Ide jön a bevezető szöveg — kommentár*)
PROGRAM végjáték (input, output);
(*Ide jönnek a típus-, változó- és
  proceduradeklarációk*)
VAR újból, felad, vége: BOOLEAN:
    válasz: CHAR;
    1:0 . . 30; (*A lépésszámláló*)
    (*stb.*)
BEGIN
    REPEAT (*Amíg az ellenfél új
            „partit" kíván kezdeni*)
      főcím; (*Procedura:
              a program bejelentkezik,
              ismerteti a játékot*)
      inicializál; (*Procedura:
                    a kezdeti értékek be-
                    állítása*)
      l:=1; (*A lépésszámláló
             kezdeti értéke*)
      REPEAT (*Amíg nincs vége
              a „parti"-nak*)
      kijelzés; (*Procedura:
      kijelzi a „parti" állását*)
      kérdez; (*Procedura:
      az ellenfelet kérdezi a gép*)
      IF NOT felad THEN
      lépésgen;
                (*Procedura: itt a
                program lényege.
                  „generálja"
                  a gép következő
                  lépését*)
                  vége:=(matt felad)
    UNTIL vége;
    kérdez2; (*Procedura: azt
             kérdezi, hogy új partit
             kezdjenek-e*)
    UNTIL NOT újból; (Amíg az
                      ellenfél már
                      nem kívánja
                      újból kezdeni*)
  elköszön (*Procedura*)
```

(3) Az adatok jó ábrázolása fontos: szoros összefüggés van ugyanis az adatstruktúra és az algoritmus között. Jól megválasztott adatstruktúra általában egyszerűbb, áttekinthetőbb algoritmust eredményez. Az egyik lehetséges kiindulás, hogy a sakktáblát ábrázoljuk a gépen, például így:

```
TYPE
  figura=(üres, saját király,
          saját bástya, ellenkirály);
VAR
  sakktábla: ARRAY ['a'..'h', 'l'..'8']
                   OF figura;
```

A sakktábla tehát ebben a megközelítésben egy kétdimenziós tömb lett volna, melynek mezői részben üresek, részben a három figurát tartalmazzák.

Mivel összesen csak három figura van a táblán, nem kell a tábla valamennyi (mind a 64) mezejét figyelni, ezért a szerző inkább az alábbi adatábrázolást választotta:

```
TYPE
  koordinátatípus=(x,y);
  pozíció=ARRAY [koordinátatípus]
                OF CHAR;
  figura=(saját király,
          saját bástya, ellenkirály);
VAR
  állás: ARRAY [figura] OF pozíció;
```

Az, hogy a pozíció *karaktereket* tartalmazó tömb, lehetővé teszi, hogy egy-egy figura pozícióját a szokásos módon adjuk meg. Például így: e2. Ez a választás azonban azzal is jár, hogy a lépés a saját királlyal („y") irányban előre így néz ki:

```
  állás [saját király, y]:=SUCC(állás[saját
        király, y]);
```

Ha a pozíciót így definiáltuk volna:

```
  TYPE
    pozíció=ARRAY[koordinátatípus]
      OF INTEGER;
```

akkor úgy léphettünk volna;

```
állás   [saját király, y]:=
        állás[saját király, y]+1;
```

Mindkét választásnak vannak előnyei és hátrányai. Ennek taglalásától most eltekintünk. A SUCC és a PRED a PASCAL két „beépített" függvénye, melyről itt elég annyit tudni, hogy az argumentumként megadott karakter után következő, illetőleg előtte lévő karaktert adja vissza.

(4) A program lényege a gép soron következő lépésének meghatározása: a lépés generátorprocedura (szubrutin). A gép minden állásban választhat, hogy a királyával, vagy a bástyájával lépjen-e. Lépése természetesen az állástól függ. A királlyal elvileg megtehető lépések száma max. 8, a bástyával megtehető max. 14. A két figurával elvileg megtehető lépéslehetőségek száma egy-egy állásban így 14+8=22. A ténylegesen megtehető lépések száma azonban ennél általában kevesebb, mert például

— a bástya nem „ütheti" saját királyát és viszont;
— egyik figura sem léphet le a tábláról;
— a király nem adhat sakkot a másik királyának stb.

(4.1) Egyfajta játékstratégia lehetne a gép számára az, hogy minden állásban generálja mind a 22 darab elvileg lehetséges lépést, majd ezekből

— kiszűri a lehetetleneket,
— kiszűri a „pattveszélyes" lépéseket, és
— a megmaradókból kiválasztja a legjobbat.

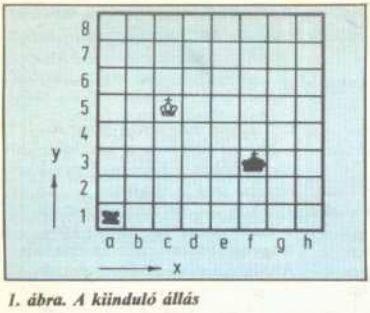
A legjobb az a lépés lenne a megmaradókból, melynél az ellenfél királyának mozgástere legjobban csökkenne.

(4.2.) Megfontolandó lehetne egy olyan megközelítés is, mely nemcsak a soron következő lépést fontolgatja egy-egy állásban és legfeljebb a „pattveszély" miatt ügyel a következő állásra, hanem a 22 elvileg lehetséges irányban elindulva, több lépésre „előre gondolkodva" választaná ki a *kereső-fa* alapján, valamilyen visszalépéses algoritmus szerint a legcélszerűbb lépést. Csakhogy ilyen megközelítésnél például akár csak három lépésre előre gondolkodni több mint tízezer lehetőség megvizsgálását vonná maga után.

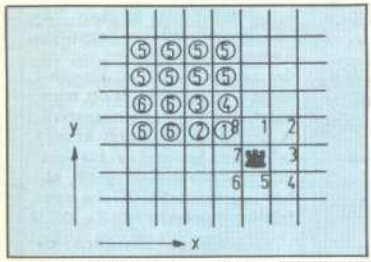
(4.3.) *A szerző egy primitívebb, harmadik utat választott:* a bástyát választotta viszonyítási alapul, és annak alapján, hogy a két király hogyan helyezkedik el a táblán a bástyához képest, minden lépésben mintegy 40-50 különböző állást különböztet meg, és minden egyes álláshoz „beprogramozta" a gép lépését.
(lásd a 2. ábrát)

Az ábrán a bekarikázott számok az ellenfél királyának, a bekarikázatlanok a gép saját királyának a bástyához viszonyított helyzetét jelölik.

A feladat megoldásának a kulcsa nyilván abban van, hogy a gép hogyan határozza meg saját lépéseit. Ez nem elsősorban számítástechnikai-programozási, hanem sakkjátéktechnikai probléma. Mindjárt az elején fontos kérdés, hogy van-e a feladatnak egyáltalán megoldása? A szerző nem sakkjátékos, de amatőr tapasztalatból tudja,



1. ábra. A kiinduló állás



2. ábra. A két király relatív helyzete a bástyákhoz képest, ahogy a „program látja"

hogy van megoldás, ha a saját királyával és bástyájával a tábla valamelyik sarkába tudja szorítani az ellenfél királyát úgy, hogy közben nem hoz létre olyan helyzetet, melyben az ellenfél királya nincs sakkban, de az ő lépése következne, és az ellenfél már nem tudna hová lépni, mert minden lehetséges helyen a gép bástyájának vagy királyának ütésébe kerülne. (Ez az ún. „pattheljzet", mely döntetlennek számít. A cél viszont: matt.)

(5) A cikk elején feltett utolsó kérdésre: (Milyen legyen az ember—gép kapcsolat?) nem lehet a konkrét géptől függetlenül megadni a jó választ. Olyan gépeken, melyekhez grafikus terminál tartozik, nyilván ki kellene rajzoltatni az egész sakktáblát, rajta a figurákkal, ki kellene jelezni a következő lépés sorszámát, jelezni kellene, ha a gép sakkot ad, jelezni kellene a mattot stb. Bár jó programnál az ember—gép kommunikáció nagyon fontos, a bemutatott programnál — többek között hely hiányában — a kapcsolatot leegyszerűsítettük: a gép szövegesen kijelzi az állást, lehetőséget ad az ellenfélnek, hogy feladja a játszmát, vagy a király új pozíciójának beadásával lépjen. A program érdekes része az a procedúra, mely azt ellenőrzi, hogy az ellenfél lépése megfelelt-e a sakkjáték szabályainak. Az ellenőrzést a szerző a programba úgy építette be, hogy ezt a procedúrát a „kérdez" procedúra hívja.

A sok előzetes megfontolás után — mert ennyi éppen belefér a cikk kereteibe — lássuk ezt a mintegy 60 sornyi „ellenőriz" nevű procedurát, mely az „XP", és „YP" karaktertípusú változóban (paraméterben) fogadja az ellenfél királyának lépését: azt a két koordinátát, amit az ellenfél megadott, és az „elfogad" nevű boolean típusú változó paraméterében megadja a hívó procedúrának, hogy az adott állásban az ellenfél lépése elfogadható-e. Az „állás" a program egészére deklarált ún. globális változó, melyhez természetesen az „ellenőriz" procedúra is hozzáfér.

A procedúra sorban a következőket vizsgálja:

— az ellenfél olyan koordinátákat adott-e meg, melyek a sakktábla koordinátái (például a procedúra nem fogadja el a „w9" koordinátapárt);

— az ellenfél ellépett-e a helyéről (azaz a beadott koordinátapár különbözik-e az ellenfél királyának beadás előtti koordinátáitól);

— lépése megfelel-e a királyra vonatkozó lépésszabályoknak (nem akart-e valamelyik irányban egymezőnyinél többet mozdulni);

— nem lép-e a gép bástyájának ütésébe (itt a gép azt is vizsgálja, hogy a gép királya nem takarja-e a bástya ütés-vonalát);

— nem lép-e a másik király ütésébe.

A program felépítése olyan, hogy ha az egyik feltétel nem teljesül, akkor a gép a többi feltételt már nem vizsgálja.

A választott adatábrázolási mód természetesen kihatott az ellenőrzések módjára. A procedúra elején SET-típusú változókat deklarálunk, melyeket az ellenőrzés megkezdése előtt az állás alapján feltöltünk. Például azt, hogy az ellenfél betartotta-e a királyra vonatkozó lépésszabályokat, a procedúra úgy vizsgálja, hogy feltölti az „EKXS" és az „EKYS" változót az ellenfél királyának „környezetével" és akkor fogadja el az ellenfél lépését, ha a megadott koordináták „benne vannak" ebben a „környezetben",

Így:

```
elfogad:= [elfogad AND (Xp IN ekxs)
           AND (yp IN ekys)];
```

A továbbiakban nem megyünk bele a részletekbe, jó fejtörést kívánunk.

(Megjegyzés: a géphez kapcsolt nyomtató csak nagybetűkkel tudott írni.)

(folytatjuk)

—KE—

```
(*************************************************************)

(*A "VÉGJÁTÉK" NEVŰ PROGRAM-BAN A GÉP KIPRÓLLYAL ÉS BÁSTYÁVAL JÁTSZIK
          AZ ELLENFÉL KIRÁLYA ELLEN*)

(*************************************************************)
(*************************************************************)

(*******)      PROGRAM VEGJATEK(INPUT,OUTPUT);      (*****************)

(*************************************************************)

TYPE   KOORDT=(X,Y);(*A SAKK-TÁBLA KÉT KOORDINÁTÁJA*)
       XSETTIP=SET OF 'Ω'..'Ä';(*A TARTOMÁNYA *)
       YSETTIP=SET OF '1'..'8';(*Y TARTOMÁNYA *)
       BETUT:SET OF 'A'..'H';(*A KEZELÉS "BARÁTSÁGOSABBÁ" TÉTELÉHEZ*)
       LEPEST=0..70;(*A LÉPÉS-SZÁM-VALTOZÓ TIPUSA *)
       POZT=ARRAY[KOORDT] OF CHAR;
            (* EGY-EGY FIGURA HELYÉT TARTALMAZÓ KOORDINÁTA-PÁR TIPUSA *)
       FIGURAT=(SK,SP,EK);(*A HAROM FIGURA:SAJÁT KIRÁLY ÉS BÁSTYA,PLE
                           AZ ELLENFÉL KIRÁLYA*)
       ALLAST=PARRAY[FIGURAT] OF POZT;(*A HÁROM FIGUFA ÁLLÁSA*)
       SMPNT=(SAKK,PATT,PATT,NORMAL);

(*************************************************************)

VAR    LILEPEST;(*A LÉPÉS-SZÁMLÁLÓ*)
       XSET:XSETTIP;   YSET:YSETTIP;
       XSET: BETUT;
       KS,BS,KE:POZT;(*A HÁROM FIGURA POZÍCIÓJA*)
       SMPNT:SMPNT;(*EZ A VÁLTOZÓ TARTALMAZZA, HOGY SAKK,...VAN-E*)
       ALLAS:ALLAST;(*AZ ÁLLÁS-T TARTALMAZÓ VÁLTOZÓ*)
       JATÉK:ARRAY[LEPEST] OF ALLAST;(* A JÁTÉK EGÉSZ MENETE,
                 AZAZ AZ EGYMÁST KÖVETŐ ÁLLÁSOK*)
       VEGE:BOOLEAN;(* VEGE A "PARTI"-NAK:FELADTA,MATT *)
       FELAD:BOOLEAN;(*A JÁTÉKOS FELADJA-E A JÁTÉKOT*)
       UJBOL:BOOLEAN;(*A JÁTÉKOS KÍVÁN-E UJABB "PARTIT" JÁTSZANI*)
       VALASZ:CHAR; (*SEGÉD-VÁLTOZÓ A DIALOGUSHOZ*)
 (*************************************************************)

PROCEDURE ELLENOPIZ(XP,YP:CHAR;VAR ELFOGAD:BOOLEAN);
(* EZ A PROCEDÚRA "XP" ÉS "YP"-BEN FOGADJA AZ ELLENFÉL KIRÁLYÁNAK
   LÉPÉSÉT                    ÉS AZ "ELFOGAD" VÁLTOZÓBAN
   VISZI,HOGY AZ ELLENFÉL LÉPÉSE ELFOGADHATÓ-E *)

VAR EKXS :XSETTIP; EKYS :YSETTIP;
    SKXS :XSETTIP; SKYS :YSETTIP;
    SDXS :XSETTIP; SDYS :YSETTIP;
    XE,XS,XP:'A'..'H'; YE,YS,YP:'1'..'8';
    TAKARX,TAKARY:BOOLEAN;

BEGIN
      EKXS:=[]; EKYS:=[];
      SKXS:=[]; SKYS:=[];
      SDXS:=[]; SDYS:=[];
      XE:=ALLAS[EK,X];  YE:=ALLAS[EK,Y];
      XS:=ALLAS[SK,X];  YS:=ALLAS[SK,Y];
      XP:=ALLAS[SP,X];  YP:=ALLAS[SP,Y];

(* INNEN KEZDVE SORBAN VIZSGÁLOM A FELTÉTELEKET *)

      ELFOGAD:=((XP IN XSET) AND (YP IN YSET));
       (* AZ ELLENFÉL NEM AKART LELÉPNI A TÁBLÁRÓL *)

        IF ELFOGAD THEN BEGIN
        (*EGYÉBKÉNT NEM VIZSGÁLOM TOVÁBB A FELTÉTELEKET*)
        ELFOGAD:=(ELFOGAD AND ((XE<>XP) OR (YE<>YP)));
        (* AZ ELLENFÉL ELLÉPETT A HELYÉRŐL *)

         IF ELFOGAD THEN BEGIN
         (*EGYÉBKÉNT NEM VIZSGÁLOM TOVÁBB A FELTÉTELEKET*)
         (*FELTÖLTÖM AZ "EKXS(ET)" ÉS "EKXS(ET)" VÁLTOZÓT *)
          IF (XE >'Ω') THEN EKXS:=EKXS+[PRED(XE)];
          EKXS:=EKXS+[XE];
          IF (XE<'Ä') THEN EKXS:=EKXS+[SUCC(XE)];
          IF (YE>'1') THEN EKYS:=EKYS+[PRED(YE)];
          EKYS:=EKYS+[YE];
          IF (YE<'8') THEN EKYS:=EKYS+[SUCC(YE)];

         ELFOGAD:=(ELFOGAD AND (XP IN EKXS) AND (YP IN EKYS));
         (*LÉPÉSE MEGFELELT A KIPÁLYRA VONATKOZÓ LÉPÉS-SZABÁLYNAK*)

          IF ELFOGAD THEN BEGIN
          (*EGYÉBKÉNT NEM VIZSGÁLOM TOVÁBB A FELTÉTELEKET*)
          IF ((XS<XP) AND (YS=YE)) THEN TAKARX:=TRUE
                                   ELSE TAKARX:=FALSE;
          IF ((YS>YP) AND (XS=XE)) THEN TAKARY:=TRUE
                                   ELSE TAKARY:=FALSE;
(* A SAJÁT KIRÁLY NEM TAKARJA-E A BÁSTYA ÜTÉSVONALÁT *)
          IF ((XP=XS) AND (NOT TAKARY)) THEN ELFOGAD:=FALSE;
          IF ((YP=YS) AND (NOT TAKARX)) THEN ELFOGAD:=FALSE;
          (* AZ ELLENFÉL NEM LÉP-E A BÁSTYA ÜTÉS-VONALÁBAN *)

           IF ELFOGAD THEN BEGIN
           (*EGYÉBKÉNT NEM VIZSGÁLOM TOVÁBB A FELTÉTELEKET*)

           (*FELTÖLTÖM AZ "SKXS(ET)" ÉS "SKYS(ET)" VÁLTOZÓT *)
           IF (XS >'Ω') THEN SKXS:=SKXS+[PRED(XS)];
           EKXS:=EKXS+[XE];
           IF (XS<'Ä') THEN SKXS:=SKXS+[SUCC(XS)];
           IF (YS>'1') THEN SKYS:=SKYS+[PRED(YS)];
           SKYS:=SKYS+[YS];
           IF (YS<'8') THEN SKYS:=SKYS+[SUCC(YS)];

           ELFOGAD:=(ELFOGAD AND NOT((XP IN SKXS) AND (YP IN SKYS)));
           (*AZ ELLENFÉL NEM LÉPETT A GÉP KIRÁLYÁNAK ÜTÉSÉBE *)

            IF ELFOGAD THEN BEGIN WRITELN(' ')
            END ELSE WRITELN('A MÁSIK KIRÁLY ÜT!')
            END ELSE WRITELN('A BÁSTYA (IS) ÜT!')
       END ELSE WRITELN('A KIRÁLY CSAK VALAMELYIK SZOMSZÉDOMEZORE LÉPHET')
         END ELSE WRITELN('NEM LÉPETT EL A HELYÉRŐL')
         END ELSE WRITELN('LE AKAR LÉPNI A TÁBLÁROL');
      IF (XP IN ['A'..'H']),THEN
    WRITELN('VAGY,NAGY BETŰVEL ADTA MEG A LÉPÉST. KIS BETŰVEL KELL');
    END; (*ELLENŐRIZ *)
```

Barátunk, a számítógép

Miért is tartalak?

(Motto: A felesleges munka kis híján egyenlő a restséggel. — Abraham Lincoln)

Picit hosszabb feldolgozást futtattok. Tudom, de mégis türelmetlenül dobolok ujjaimmal az asztalon. Most kicsit hangosabb kotyogást érzek — jön az eredmény. Kotyogás? A szerelő szerint ez baj. Nyugi, öregem, nincs itt baj. Látod a szaporább fényt? A vincseszter most keményebben dolgozik — ezt tudom. Ezért érzem a picivel erősebb vibrálást: jön az eredmény.

Furcsálják, hogy barátunknak miért a külsejét mutattam be először? Több okom is van rá. Nem távoli nagygépről van szó, hanem mindennapi eszközünkről. Ha kényelmetlen a szék, az asztal, akkor nem jól dolgozunk. A fal színe sem közömbös. Pont barátunk külseje ne számítana? És vajon tartós barátságaink nem a külső jegyeken alapuló szimpátiából születtek egykor, miközben a „pusztán" lelkirokonnal nehezen találjuk meg a hangot, mert valamely külső vonását nehezen fogadjuk el? Ezelőtt 20 évvel mutattak nekem egy számítógépet és annak Autocode nevű (óh, borzalom!) programozási kézikönyvét. Annak látása előtt akár még programozói jövő is állhatott volna előttem . . .

Most viszont a külcsín helyett a belbecsre fogok koncentrálni. Néhány tipikusnak mondható példán keresztül megnézzük, hogy általában mire való, miként működnek együtt „testrészei", és mikor célszerű a számítógép használata.

Mikor használjuk jól kis barátunkat, és mikor nem?

Az „értelmes" írógép

Kis hazánkban sajnálatos tapasztalat, hogy a számítógépet igen sokan író gépnek tekintik. Ki ne hallott volna még az x példányos tablók tonnáiról(!), amelyek y számú vetületben készülnek? Közüzemi szolgáltató vállalatunk számítóközpontjában jó tíz éve azért terveztek át egy rendszert, mert sok papírt fogyasztott. A dicséretes példával milliókat takarítottak meg. Ez azonban még nem általános. Számítógépeink ontják a „tablókat". Pedig tudni illenék, hogy:

— Ha valahol tablókat gyártanak, akkor ott igen alacsony fokúan dolgozzák fel az adatokat. Géppel. Majd emberek hadai próbálják meg kivenni a tablókból a lényeget. Bármely hárompéldányosnál több példányos tabló bűn a számítógép és az emberiség ellen. Számítástechnikai bontott csirke, amit még a felhasználónak kell megsütnie — ha tudja.

— A rendezés és listázás a legprimitívebb feladat, amit egy számítógépre lehet bízni. Működnek a háttértárak, kikészül a nyomtató. Barátunk agyát, a központi egységet lefoglaljuk ezzel a primitív feladattal. Ha tehetné, sűrű ásításokat nyomna el ilyesfajta feldolgozások közben.

— Kis barátunk, a mikro egyik sebezhető pontja a rendezés. Bár léteznek már viszonylag gyors rendezőrutinok is, mikrón csak ténylegesen végső szükség esetén illik nagy adattömegeket rendezgetni. A mikrókhoz kapcsolt nyomtatók többsége pedig nem a többpéldányos, nagy tömegű adatkijelzésekhez készült.

Furcsa. Ha valamelyik barátom többet nyújt, mint amit várok, és másként, amint megszoktam, akkor haragszom rá. Vannak, akik azért idegenkednek a személyi számítógépektől, mert azokon nem lehet végtelenül és boldogan „tablózni". No, igen! Egy tabló felett el lehet mélázni azon, hogy tulajdonképpen milyen információt is kellene, lehetne kihámozni belőle. Ha nincs tabló — de van azonnali lekérdezhetőség —, akkor előre tudni kellene információinknak szerkezetét. Annál pedig nincs sértőbb, amikor az embert barátja akarja jó útra téríteni.

Beszélgetőpartner

Korlátos — nem korlátolt — írásbeli kijelzési képességei miatt a mikrók igazi barátjaivá azok váltak, akik szeretik a párbeszédet. Lám, valamit közlök kis barátommal, és ő azonnal reagál. Nem kell papírra várnom. Nem kell a gépteremben leadnom a „dzsobot". Beütök valamit, mire barátom a háttértáron levő vagy az onnan a központi egységbe töltött ismereteken eligazodva válaszol a képernyőn. „On-line", „interaktív", „dialógus". Remek lehetőség ez, de . . . Illenék tudni, hogy:

— Barátunk azonnali válaszai sok esetben olyanok, mint egy esti, borközi állapotban lefolytatott beszélgetések esetén adott baráti tanácsok. Igen jóindulatúak, de mélységben nem mindig átgondoltak. Nem feltétlenül a követendő utat jelentik. Fellelkesülve a személyes kapcsolattól, elnézést a kifejezésért, sokszor magunk is hülyeséget válaszolunk. Olyasmit, amit levélben sohasem írtunk volna le. Ami félrevezeti a „beszélgetés" fonalát. Tudnunk kellene tehát mérsékelnünk magunkat. Arra gondolnunk, hogy ott van a nyomtató. Majd a reggeli napfénynél, friss fejjel átgondoljuk a hirtelen tanácsokat.

— Barátunk olyan, mint számos személyes barátunk. Kérdés, válasz, párbeszéd. Lám, eltelt egy este és mégsem váltottunk okos szót! Mindketten ismételtük magunkat. A mikrogépek látszólagos hatékonyságának bűvöletébe estünk. Mert ő szószátyár. Közli velem, hogy itt tart, azt csinálja, legyek nyugodt . . . Fél perc eltelik, mire eredményt kapok, de közben elszállt az idő, mintha régi kedvesünkkel idéztük volna fel a múltat. Ezalatt egy nagygépen harminc hasonló feldolgozást, tranzakciót, műveletet végezhettünk volna. Nagy ámító ám barátunk! Felhasználójának igen céltudatosan és okosan kell beosztani idejét. Mintha nagygépen dolgozna.

A fejszámoló

Vess fel bonyolult számítási problémákat kis barátunknak, és találkozol Patakival! Milyen jó, hogy valamely problémánk megoldására a feldolgozás közben a képernyőn „ablakot" nyithatunk, és azon számítási műveleteket végezhetünk el. Ezt tegyék meg a nagygépesek!

Szóval listázom a készletinformációkat a képernyő egyik részén, míg a másikon — az ablakon — összeszorozhatom a mennyiséget az egységárral. Ragyogó lehetőség ez az ablak. Általában véve az ún. „spreadsheet" lehetőség. Magyar szó erre még nincs. Látsz valami információt, ami megkap. Ekkor kiteregeted — hagyományosan — a lepedőnyi papírodat (spread = kitárni, kinyitni; sheet = ív, lepedő), és ott munkálatokba fogsz. Ez nagyszerű. A számodra rendszert készítő talentuma kevésbé az. Ha valamit gyakran kell teregetni, akkor miért nem készül el a megfelelő program?

Ezekben az esetekben kis barátunk agya látszólag szorgalmasan dolgozik, miközben egyéb „tagjai" ernyedten pihennek. Nem mozgatjuk meg háttértárát, bemeneti és kimeneti egységeit. Ha csak így használnánk, akkor előbb-utóbb félresportolt egyén válna belőle. Erős karizmokkal és satnya lábakkal, vagy fordítva.

A felesleges munka

Mint látható, barátunk alkalmazása sok felesleges munkát okozhat számunkra. Hogyan? Ha nem íratnék ki x példányos tablót y számú vetületben, akkor emberek tucatjainak nem kellene emészteniük az adatokat és közben emésztődniük. Jó lenne a párbeszéd. Beszélgetek a számítógéppel? Jó élmény. Annyira, hogy észre sem veszem: eltelt az idő. Kis barátunk nem kényszerít előre megfontolt párbeszédre. Készséggel válaszol minden kérdésre. Csak éppen az idő rohan közben. Neki, a zseniális fejszámolónak, bármikor összetett számítási feladatokat adhatok. Türelmes: ha ugyanazt a feladatot hússzor adom, annyiszor oldja meg. De jó nekem ez? Csúnyán fogalmazva (nagyon csúnyán): nem lettem palira véve?

Csak az hiszi, hogy a számítógéppel, az újdonsült mikróval időt és energiát takaríthat meg, aki nem ismeri a barátok természetét. (Vagy a barátság lényegét, amelynek nem feltétlen és elsőrangú jellemzője a

minél huzamosabb együttlét. Sőt!) Kis barátunk valójában nem ideális partner. Nem olyan, mint egy jó feleség, a szó mély értelmében vett barát. Az indokoltnál türelmesebb. Elhiszi, hogy vele közölt érdekeink valódiak: nem kérdezi, nem lenne-e jobb, ha . . . Felesleges munkáktól nem óv meg. Az pedig egyenlő a lustasággal. Haszontalan.

Egyensúly

Nem árulom el, hogy a számomra ma is legdrágább lény megpillantásakor mi ragadott meg engem. Más férfi máshová pillantott volna. Jellemző ránk, emberekre, hogy egyetlen lényeg felé próbálunk figyelni. Pontosabban: lényegnek tartott dolog felé.

Így előfordulhat, hogy veszünk, alkalmazunk egy számítógépet úgy, hogy annak egyes tagjai — használat híján — elzsibbadnak. Miközben túlterheljük egyik egységét feladatokkal, a többi — képességeihez mérten — kihasználatlan marad. Valójában kis barátunk akkor éri meg a pénzét, ha moderáltan értelmes írógépként, józanul korlátozott beszélgetőpartnerként, és nem ismétlődő csodás fejszámolóként egyidejűleg alkalmazzuk. Természetesen adott környezetben valamelyik feladat dominálhat. Remélhetőleg ott megfelelő konfigurációt, testelrendezést alkalmaznak. A tipikus alkalmazási környezetben azonban a számítógép egységeinek kellőképpen egyenlő szintű terhelésére van szükség. Ehhez még nem szoktunk. Az esetleg távoli idegen, a nagyszámítógép, rossz társas szokásokat generált bennünk. A közelebbi, bár kisebb baráthoz sokan még a másként szokott módon próbálunk közelíteni. Nem találva az egyensúlyt.

Zárszóként. Egyszer volt egy előkelő, nagy hatalmú idegen — a számítógép —, „aki" csak kevesek barátjává vált. Most pedig itt van a kezes jó barát, a mikro. Fantáziadús ifjú. Sokra képes. De az érett felhasználónak kell megtalálnia, felnőttként, a megfelelő egyensúlyt, a baráti összhangot. E téren fontos az úgynevezett „szakma" szerepe. De erről majd legközelebb.

DR. HALASSY BÉLA

Hasznos tanácsok floppy-tulajdonosoknak

Ha megszavaztatnánk a mikroszámítógép-felhasználókat, valószínűnek majdnem mindenki a floppyt tartaná a legjobb háttértárolónak, mert elég gyors, olcsó és megbízható.

Az olcsó magnókazettás rendszerek túl lassúak, szolgáltatásaik nem érik el a legegyszerűbb lemezes egységekét sem, és általában nem eléggé megbízhatóak. A merevlemezes egységek (Winchester) nagy mennyiségű adat tárolására alkalmasak és gyorsak is, de természetesen nem tekinthetők olcsó megoldásnak. Ez a magyarázata annak, hogy a mikrogépek területén a hajlékony mágneslemezes tárolóknak kitüntetett szerepük van.

Néhány szakértő szerint az olyan mikrogép, amelynek csak kazettás egysége van, idejének nagy részét „várakozással" tölti; arra vár, hogy floppyt illesszenek hozzá. Természetesen van néhány olyan alkalmazási terület, ahol a floppy csak luxus, de az esetek többségében „tisztességes" számítások, adatkezelések csak akkor végezhetők el, ha legalább egy, de inkább két floppy is működik a rendszerben. Sajnos a hazai mikrogépek és a hozzájuk illesztett floppyk esetén az ár lehet elriasztó tényező is, és üzemeltetésük sem problémamentes.

A továbbiakban igyekszünk rávilágítani néhány üzemelési probléma forrására, és javaslatot teszünk a hibák elhárítására.

A floppyról általában

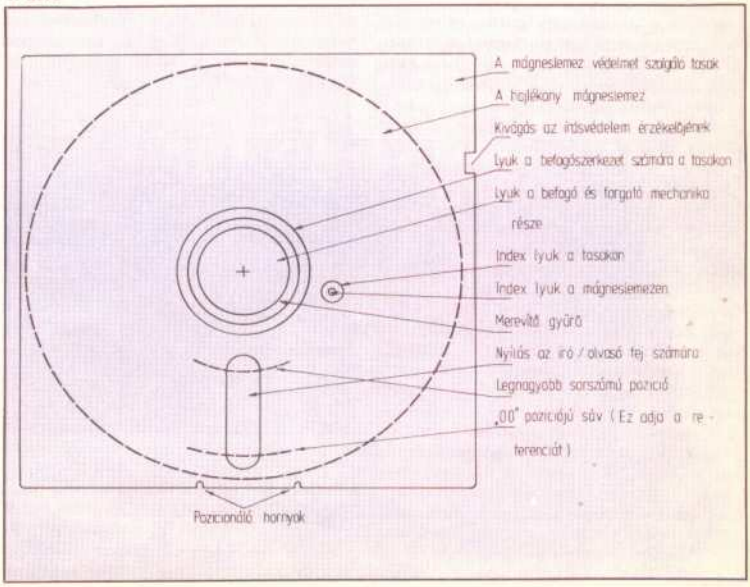
Vizsgálatunk tárgyául az 5 1/4"-os minifloppyt választottuk, mert ez ma a legelterjedtebb. Az új tervezésű rendszerek 96 sáv/inch és dupla sűrűségű felvételt is lehetővé tesznek, de a piac nagyobb részét ma még a 48 sáv/inch meghajtók foglalják el.

A tervezők sokat foglalkoznak a kompatibilitási probléma megoldásával, amely az új, nagy jelsűrűségű lemezek működtetésére épült meghajtók alacsony jelsűrűségű lemezekkel történő használatakor merül fel. Ezeknél az új lemezegységeknél a kezelő „szoftvert" közvetlenül a meghajtó elektronikája mellé építik be, és segítségével a rendszer automatikusan felismeri a különböző jelsűrűségű lemezeket.

Mindenekelőtt nézzük meg röviden, hogyan működik a floppy. Az adatok mágneses úton kerülnek a hordozó alapanyagra. A rögzítés elve bizonyos mértékig hasonló a mágnesszalagos egységeknél megszokottakhoz, eltekintve a nyílvánvaló mechanikai különbségektől. A kör alakú mágnesezhető lemez forog, ami a szalag mozgásának felel meg. Az író/olvasó fej pedig sugárirányban „szabadon" mozog.

Az adatok a mágneslemezen koncentrikus körökön helyezkednek el, melyek bármelyikét könnyen el lehet érni. Nézzük egy pillanatra a hordozót *(1. ábra)*. Ez egy vékony, hajlékony, mágneses bevonattal ellátott műanyag lemez, amely szintén hajlé-

1. ábra

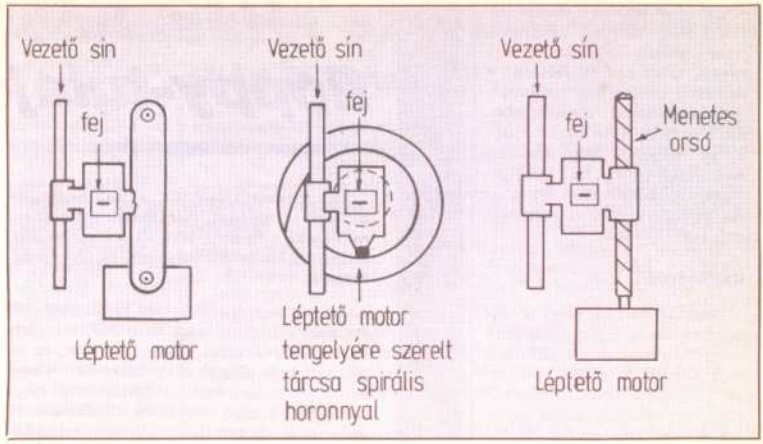


kony tasakban foglal helyet. A lemez alatt és felett kb. 0,2 mm vastag műanyag „szivacslap" van, ami a könnyebb futást segíti elő. A lemez vastagsága gyártótól függően 0,1—0,15 mm. A tasakon néhány nyílás látható, amelyeken keresztül a lemezre írt információ elérhető. Egy kapcsolómechanizmus a középső nagyméretű kör alakú nyíláson át fogja meg és rögzíti a tengelyhez a lemezt. Ez a kapcsolómechanizmus forgatja a lemezt a tasakban 300 ford/perc, 8" lemeznél 360 ford/perc sebességgel. Nem is olyan egyszerű dolog ez! Próbáljuk meg a kézben tartott lemezt megforgatni a tokban. Érezhetően nehezen mozdul! Ám a piciny motor a megfelelő kialakítású befogószerkezet segítségével könnyedén viszi, pörgeti a lemezt a tasakban.

A leggyakoribb mechanikai hibák

Itt adódik az első probléma, a megfelelően pontos, centrikus, gyűrődésmentes lemezbefogás. Mialatt a vékony műanyag lemez, amelyen kb. 10—30 mikron vastagságú „mágneses fóliaréteg" van, forog a tasakban, egy kemény, keramikus fejet hozunk vele érintkezésbe, amely a jeleket a felületre felírja, illetve leolvassa. Az író/olvasó fej kb. 2-3 gramm súlynak megfelelő erővel nehezedik a forgó lemezre. VC—1541-nél az érték 5 gramm. Az egyoldalas kialakításnál a fejjel szemben, a lemez másik oldalán egy filckorong támasztja meg a lemezt. A kétoldalas kialakításnál azonos felépítésű fej van a másik oldalon is. A fej, illetve filc és a lemez között jelentős a súrlódás, ami a mágneses bevonat kopását eredményezi, és a fej elszennyeződik. A támasztófilcet időnként cserélni kell.

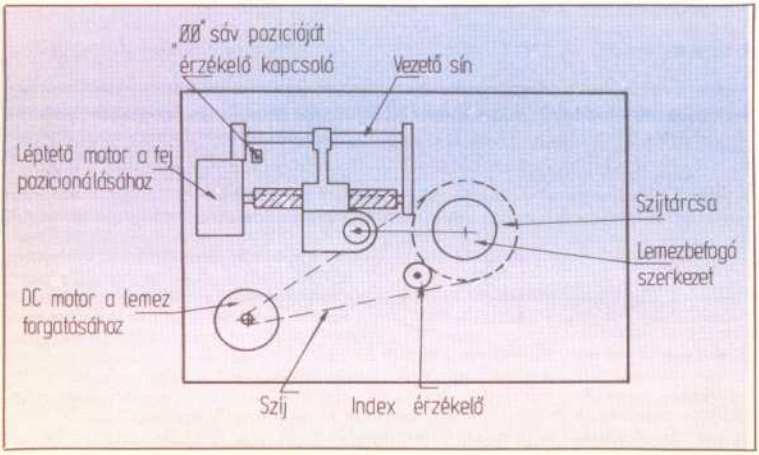
A floppyk, tekintet nélkül származási helyükre, lényegében azonos felépítésűek. Egy tipikus kialakítású floppyt láthatunk a *2. ábrán*. A lemezt rögzítő mechanizmust az olcsóbb megoldású egységeknél lapos szíj hajtja meg a sarokban elhelyezett motor segítségével. Ez ma már nem tekinthető korszerű megoldásnak. A szíj kilazulhat, csúszhat, nem megfelelő anyag esetén sztatikus feltöltődés forrása lehet. Ez a hiba néhány korai modellnél jelentkezett. Ezek a problémák általában csak huzamos használat után jönnek elő, és a szíjszakadás is viszonylag ritka.



3. ábra

A meghajtó motor forgása vagy a hálózati frekvenciához van szinkronizálva, vagy egy külön fordulatszám-szabályozó biztosítja az állandó sebességet. A fordulatszám stroboszkóppal ellenőrizhető. A strobotárcsát rendszerint a motor tengelyére rögzítik. A néhány százalékkal lassúbb vagy gyorsabb forgás ritkán okoz problémát, mert a jelfelismerő áramkörök bizonyos mértékig sebességtoleránsak. Sokkal több probléma származik a gyors sebességváltozásokból, az egyenetlen forgásból. Az eredmény hibás olvasás lesz, ennek ellenére vizsgálatkor az olvasó elektronikát hibátlannak találjuk. A hiba oka lehet például az elszennyeződött szíj, a fordulatszám-szabályozó elektronika lengései, a megkopott motorcsapágyak, a hibás tápegység, IC vagy egyenirányító.

2. ábra



Az új típusú, lapos profilú, kefe nélküli, DC motorral közvetlenül meghajtott, ún. direct-drive egységeknél már elmaradnak a szíj által okozott problémák, egyenletesebb a futás, kevesebb a csapágy, nincs szíjtárcsa stb. A mechanikus alkatrészek számának jelentős csökkentése nagyobb megbízhatóságot eredményez jobb helyikihasználás mellett. Mindezek mellé speciálisan tervezett IC-k, berendezésorientált áramkörök kerültek, és így a vezérlő elektronika is egyszerűbb, megbízhatóbb lett, a startidő lecsökkent kb. 0,5 másodpercre.

Az író/olvasó fejet egy műanyagból vagy lemezből kialakított hordozókar tartja, amely egy vagy két vezetősín által meghatározott pálya mentén, a lemezsugár irányában szabadon elmozdulhat. A csúszómechanika könnyen bepiszkolódik, ami a fejhordozó kar megszorulásához vezethet, hibás léptetést (fejpozicionálást) eredményezhet. A probléma megoldása a csúszósínek alapos tisztítása. Ügyeljünk arra, hogy a tisztítószövetből szálak ne maradjanak vissza a sínen. Néha célszerű lehet a tisztításhoz néhány csepp alkoholt is használni.

Gyártmánytól függ, hogy a sínek kenésére kenőanyagot használjunk-e. Ez ugyanis további szennyezéseket gyűjt, ha lehet mondani, még többet és gyorsabban. Ha a kenőanyag használata a megfelelő működés érdekében elkerülhetetlen, bánjunk nagyon takarékosan vele, hogy csak oda kerüljön, ahová feltétlenül szükséges. A csúszósín kenésére jól alkalmazható a fehér vazelin, de számoljunk azzal, hogy gyakori tisztításra van szükség. További kenési helyek lehetnek a motorok csapágyai. Itt könnyű műszerolajat kell használni, de semmiképp ne használjunk sprayt. Az elszennyezi azokat a helyeket is, ahol az olaj egyáltalán nem kívánatos.

A *3. ábrán* három gyakran használt fejpozicionálási módszert mutatunk be. Mindhárom jól működik a gyakorlatban. A legpontosabbnak és legmegbízhatóbbnak a csavarmenetes mozgatást tartják. A csigavonalas pozicionálónak van néhány problémája, amiről érdemes tudni. A csiga-

vonal pályáról a vezető kiugorhat, ha a lemezegységet rázkódás éri, például szállításkor, vagy hibás vezérléssel a fej könnyen kifuthat a spriálpálya végén. Gyakran nincs más mód a javításra, mint kinyitni és visszatenni a helyére a pozicionálót. Normális esetben a pozicionálótárcsát néhányszor körbeforgatva, a pozicionálókar önmagától visszatér a helyére.

Újabb kivitelű lemezeknél, amikor a meghajtó éppen nincs használatban, az író/olvasó fej nincs a lemezhez szorítva, így a lemezt forgató motor néhány másodperces késleltetés után leáll, ezzel a mágneses adathordozó anyag kopása is csökkenthető. A vezérlő elektronika biztosítja, hogy intenzív lemezhasználat esetén a motor nem áll le, és várakozással nem tölt sok időt a rendszer.

Az író/olvasó fej le- és felemelését biztosító elektromágnes és a hozzá tartozó feszítőrugók egy sor problémát okozhatnak. A leégett tekercs nyilvánvalóan nem működik, de egy erősebben meghúzott feszítőrugó mellett a rendszer hibátlanul ír/olvas, mialatt jelentősen koptatja az információt hordozó mágneses anyagot. Ha túlságosan laza a rugó, a fej el-elhagyja a lemezt, és így ingadozó jelszintet eredményez, ami természetesen hibás működéshez vezet.

Az egyoldalas lemezmeghajtó egységeknél a szennyeződés és kopás miatt rendszeresen cserélni kell a fejjel szemben lévő oldalon a támasztófilcet.

A kétodalas meghajtók különösen érzékenyek a rugófeszítő erőre. Ha ugyanis túlságosan erős a rugó, akkor a két fej egyszerre olyan erővel üti meg a lemezt, hogy a mágneses réget megsérülhet. Néhány régebbi megoldásnál állandó kontaktus van a fej és a forgó, mágneses adathordozó között.

Az index/szektor-érzékelés LED-fotodióda kialakítású. Ezt az impulzust sokféle módon használhatja a vezérlő elektronika. Az indexérzékelő legfontosabb szerepe, hogy az információt hordozó sáv kezdetét jelzi az író/olvasó elektronika számára. A szoftszektoros diszkeknél egy indexlyuk van a lemezen. Az indexpulzus érzékelőjének hibája vagy elpiszkolódása DRIVE NOT READY hibaüzenetet eredményez.

Az írásvédelem minden lemezegységen megtalálható. Az újabb kialakításoknál szinte mindenütt led-fotodióda kombinációt használnak. Ennek hibája vagy szennyeződése írási problémát okoz.

Hasonlóan a kazettás magnóhoz, a fejbeállítás szintén nagyon fontos, ugyanis a legtöbb probléma abból adódik, hogy a fej nem ott van, ahol kellene.

Gyakori hiba, hogy a fej nem pontosan radiális irányban mozog a mágneslemezen. A hiba úgy jelentkezik, hogy mindig kb. azonos sorszámú sávnál áll le olvasási hibával a lemezegység. A pozicionáló szerkezet esetleges sérülése, a műanyag megöregedéséből adódó deformáció a fej elfordulását eredményezi. Ez az eltérés rontja a frekvenciamenetet és a jel amplitúdóját is. Az eredmény hibás olvasás lesz. Az előbbi hibákat a *4. ábra* szemlélteti.

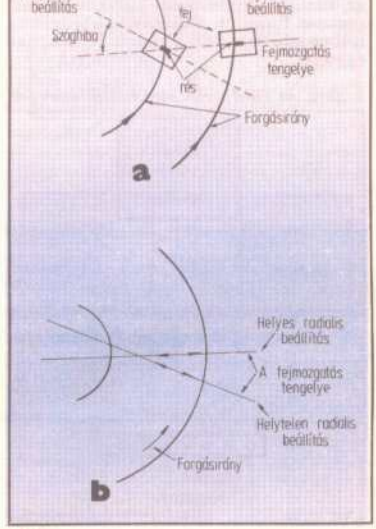
Az említett két esetben a lemezegység „saját" lemezeit elfogadhatóan kezeli, de nem lesz „kompatibilis" a többivel.

A beállításokkal együtt meg kell keresni a jelmaximumot is az egyes sávokon. Ez rendszerint a léptetőmotor házának néhány fokos elforditásával elvégezhető, mialatt oszcilloszkóppal vizsgáljuk a jelszintet az író/olvasó erősítő kimenetén. Az elfordítást óvatosan végezzük, mert általában kevesebb, mint 0,2 mm fejelmozdulás szükséges.

A három beállítást egymás után kell elvégezni, néhányszor ismételve. Sok gyártmánynál nem állítható mindhárom lehetőség, a gyári beállítást véglegesnek kell tekinteni. Ez azonban sajnos nem jelenti azt, hogy nem állítódik el. Ilyenkor az egész egységet egy gyárilag beállított újjal kell kicserélni.

Az elmondottakból természetesen nem következik, hogy a beállításokra hetente vagy havonta van szükség. Ezt mindenkor a gyakorlat mutatja meg.

Végül még egy nagyon fontos karbantartási feladat a fej tisztítása. Hasonlóan a magnókhoz, a fej elpiszkolódása a jel gyengüléséhez, az átvitel romlásához vezet. Célszerű rendszeresen tisztítani. Néhány nagyon makacs szennyeződés a tisztítólemezzel sem távolítható el. Ilyenkor jó szolgálatot tesz az alkoholos vatta, műanyag vagy hurkapálcán. Fémszerszámokkal óvatosan bánjunk a lemezegységek szerelésénél, a



4. ábra

fejhez soha ne nyúljunk vele! A tisztítandó felületről nagyító alatt vizsgálva távolítsuk el a szennyeződést. A legtöbb esetben az olvasási hiba eltűnik, és a rendszer minden mechanikai utánállítás vagy csere nélkül „életre kel". Csak akkor kezdjünk a mechanikai utánállításhoz, ha a tisztításoknak nincs kielégítő eredményük.

Az író/olvasó fej kialakításától nagymértékben függ a lemez „szennyeződésre" képessége. A régebbi megoldásoknál alkalmazott „lencse" profilú fej mintegy belekeni a szennyezést a lemez felületébe, és így meg is sérülhet a mágneses réteg. A modern meghajtóknál a megfelelően kiképzett fej nem gyűri maga alá a szennyeződést.

Gondoljuk meg, hogy az író/olvasó fej 300 ford/percnél kb. 20 km/óra sebességgel halad át a szennyezett felületen, és fokozatosan tovább mélyíti a kialakult „barázdát". Egy sokat használt mágneslemez felületét nagyító alatt vizsgálva jól felismerhetők a porszemcsék által „vágott" barázdák. A mágneses hordozó mechanikai sérülése a felvitt adatokat is olvashatatlanná teheti. A sérült lemezekről az adatok és programok rekonstruálása csak speciálisan erre a célra írt programokkal valósítható meg.

Elektromos hibák

Ezek közül leggyakoribb a készülék túlmelegedéséből adódó probléma. A tápegységgel egybeépített gépeknél gyakori a hűtőnyílások elzárása. A trafó túlmelegedéséből és az IC-s tápegység nem megfelelő hűtéséből adódó meleg nem használ a mágneses adathordozónak sem, de a teljes berendezés élettartamát is kedvezőtlenül befolyásolja. Mindig vizsgáljuk meg a tápegységek pufferkondenzátorait, nem vesztettek-e kapacitásukból. A veszteség akár 50% is lehet. A trafó melegedése (normál hőmérséklete 40—50°C) adódhat az egyenirányító hibájából, amit szintén célszerű megvizsgálni.

Nem túl gyakori hiba, de előfordul, hogy a táp melegedés után lekapcsol. Vizsgáljuk meg a hőkontaktust, esetleg cseréljük ki az IC-t. 35—45°C a táp IC-k hőmérsékletére normális lehet. Ha a hűtőborda „hideg" és az IC vagy dióda „meleg", az rossz hőkontaktusra enged következtetni.

A Commodore lemezegységeknél gyakori hiba, hogy a fájlkeresés megindul, de további információ nem jön, és a rendszer „lefagy". A problémát korábbi gyártmányoknál sokszor a 74LS14-es IC hibája okozza. Ez az IC könnyen meghibásodik, ha a készüléket bekapcsolt állapotban csatlakoztatjuk a központi egységhez. Ügyeljünk arra, hogy a perifériákat, botkormányt csak kikapcsolt készülékhez csatlakoztassuk!

A +12 V feszültségszabályozó IC is gyakran meghibásodik a nem megfelelő hűtés miatt. Ügyeljünk arra, hogy ne zárjuk el a szellőzőnyílásokat. Rossz hűtés következménye lehet a mikroprocesszor meghibásodása is. Gyakran fellép a 74LS14-es mellett a 7406-os IC hibája is.

Az elektronika hibakeresése és javítása nem egyszerű feladat, kellő ismeretet és műszerezettséget kíván. Mindig tartsuk szem előtt, hogy a helytelen javítás nagyobb kárt okozhat a berendezésben, mint amekkora eredetileg volt.

GALINA FERENC

Az IBM helyi hálózatai

Az alábbiakban néhány megvalósított helyi hálózat ismertetésével illusztrálom az eddig elmondottakat. A szakirodalom nagyszámú helyi hálózati rendszerről tudósít, főleg nyugati típusokról, melyeket vagy irodaautomatizálási, vagy ipari folyamatvezérlési feladatkörben használnak. A helyi hálózatok terén is meghatározó eseménynek számított, amikor 1984-ben az IBM is megjelent a piacon személyi számítógépes (PC) helyi hálózatával. Ennek hatása valószínűleg meghatározza a helyi hálózatokkal kapcsolatos fejlesztések további irányát. Az újabb fejlesztések jelentős része várhatóan az IBM vonalát fogja követni mind a hardver, mind az alkalmazásfejlesztések tekintetében. Ezért a felhasználók és a tájékozódni kívánók érdeklődésére is elsősorban az IBM hálózatok tarthatnak számot.

A mai napig az IBM két irodaautomatizálási feladatkörre ajánlott hálózatot jelentett be: a Sytec-kel közös fejlesztés keretében kialakított „széles sávú" rendszerű helyi hálózatot — ez jelenleg is kapható a (nyugati) kereskedelemben —, valamint a „token ring" rendszerű helyi hálózatot, mely az előrejelzések szerint ez év végén jelenik meg a piacon.

Az IBM széles sávú helyi hálózata

A koprodukcióban kifejlesztett termékkel az IBM PC, XT, AT és „hordozható" PC gépeit lehet helyi hálózattá összekapcsolni. A hálózat 2 Mbps sebességű, vele alapkiépítésben 9, maximális kiépítésben 72 munkaállomás kapcsolható rendszerbe. A logikailag busz-, fizikailag fastruktúrájú rendszer 1 km-es, 75 ohmos koaxiális kábelen CSMA/CD hálózatelméréssel működik. A „széles sávú" megjelölés a rendszer frekvenciáthelyezéses működésmódjára utal, azaz a forgalmazni kívánó munkaállomás az 50,75 MHz-es sávban forgalmaz; ezt a jelet a hálózat frekvenciaváltó egysége 219 MHz-re „helyezi át" (vételi csatorna), és a hálózat vételkész munkaállomásai az így vett adatcsomagokat a már ismertetett módon kezelik le (címvizsgálat, címfelismerés esetén a csomag adatrészének beolvasása, nyugtázása stb.).

A rendszer adási és vételi irányban ugyanazt az adatutat (kábelt) használja, csak más-más frekvenciasávban. A megoldás előnye, hogy rendszertechnikailag magában hordozza az adatátvitellel egyidőben folytatható hang, videotex és képátvitel lehetőségét, bár jelenleg az IBM ez utóbbihoz még nem ajánl hardvert (csatolóegység), hanem ehelyett a Syteckel való közvetlen kapcsolatfelvételt javasolja. A helyi hálózati hardvert ugyanis a Sytec fejlesztette ki.

A hálózat MS—DOS 3.1 operációs rendszer alatt működik. A rendszerben fájlok és üzenetek továbbíthatók a hálózat egyes gépei között, és lehetőség van a konkurens módon történő osztott lemez- és nyomtatókezelésre is.

Részegységek

A széles sávú helyi hálózat az alábbi részegységekből áll:

— IBM PC Network Adapter (hálózati csatolóegység),
— IBM PC Network Cable Kit (kábelezési elemkészlet),
— IBM PC Network Translator Unit (fekvenciaváltó egység).

A hálózat felépítését, az egyes hálózati részegységek szerepét és egymással való kapcsolatát az *1. ábra* szemlélteti. A kábelezési elemkészlethez tartozik a 8 állomásos kábelcsatlakozó, melynek segítségével max. 8 db PC kapcsolható rendszerbe. A maximális telepítési távolság alapkiépítés esetén 200 láb. A bővített kiépítésű rendszerben az egyes munkaállomások a frekvenciaváltótól 1, 400, illetve 800 láb távolságra telepíthetők a választható kábelhossztól (rövid, közepes vagy hosszú kábelszakasz) függően. A fenti konfigurációban a PC-k közötti távolság tehát szélső esetben 1000 láb is lehet.

A csatolóegység

A széles sávú hálózati csatolóegység egykártyás nyomtatott áramköri kivitelben készül, nagy integráltságú áramköri elemeket tartalmaz (mikroprocesszor, adatátvitel-vezérlő, nagy kapacitású RAM/ROM tárak stb.), és tárolt mikroprogramja, valamint önálló feldolgozókapacitása révén — főleg az adatátvitel-vezérléssel közvetlenül kapcsolatos funkciók ellátásával — nagymértékben tehetmentesíti a csatolt PC központi egységét.

A csatolóegység fizikailag a PC belső sínrendszerét képező rendszerinterfész és a helyi hálózati koaxiális kábel közé kapcsolódik és a PC-be kell bedugaszolni. Blokkvázlatát a *2. ábra* szemlélteti.

A csatolóegység feladatai:

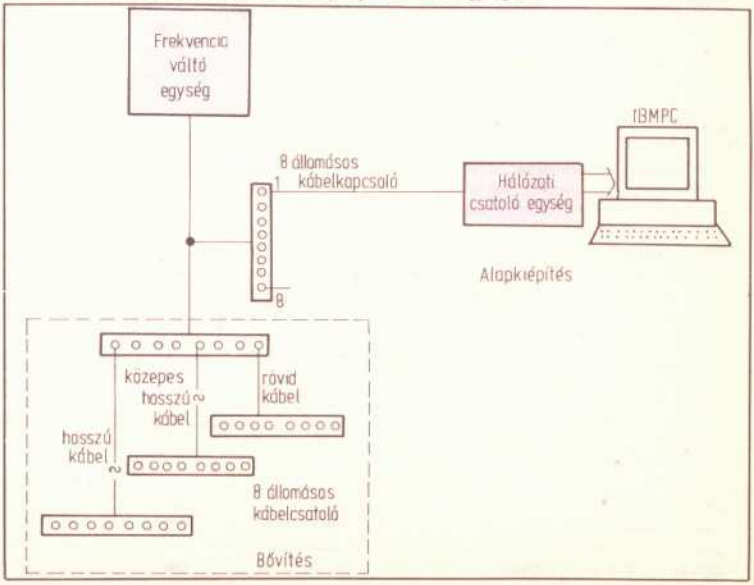
— adatok „csomagokká" szervezése (kiegészítő információk hozzákapcsolása, például cím),
— vonalfoglaltság és csomagütközés figyelése,
— adatcsomagok továbbítása az 50,75 MHz-es csatornán,
— adatcsomagok vétele 219 MHz-en, címfigyelés, adatrész beolvasása,
— adatpufferelés,
— DMA-kezelés.

Az adatátvitel szervezése

A csatolóegység a kártyára épített hardver, illetve az ott ROM-ban tárolt mikroprogramok segítségével az OSI nyílt adatátviteli architektúra két rétegéből ötöt valósít meg. Ezzel olyan alkalmazói interfészt hoz létre, amely mentesíti az alkalmazásfejlesztőket az adatátvitel szabályainak részletes ismeretétől — mint ahogy a telefonáló sem „tudja", hogyan működik a telefonközpont.

A csatoló az operációs rendszertől (MS—DOS, CP/M 86, XENIX, UNIX stb.) nagymértékű függetlenséget biztosít. A csatolóegységbe beépítették a rádiófrekvenciás modemet is, amely megvalósítja az 50,75 MHz-en való adást és a 219 MHz-en történő vételt. Az Intel 82586-os adatátvitel-vezérlő és a Sytec soros interfészvezérlő áramkörei valósítják meg az OSI architektúra fizikai és adatkapcsolati rétegfunkcióit

1. ábra. Az IBM széles sávú hálózatának felépítése és részegységei



(adatcsomagok ütközésének figyelése, rádiófrekvenciás alapsávú digitális jelátalakítás, adatok csomagokká szervezése stb.), a további rétegfunkciót (hálózati, szállítási viszony) az Intel 80188 processzor és a környán tárolt mikroprogramok segítségével szoftverben valósították meg.

A csatolóegység és a PC közötti rendszerinterfész forgalmát a PC interfészvezérlő áramkör és a hálózati be/kivitel vezérlő programjai (NETBIOS) vezérlik DMA (direkt memóriahozzáférés) segítségével.

A hálózat címkezelési rendszere rugalmas. Az egyes állomásoknak kétféle címük van: egy állandó, ROM-ban tárolt fizikai és egy 1-től 6-ig terjedő logikai címük, amit a csatolón RAM-ban tárolnak. E címzési rendszerrel, a csoportcím megadásával az üzenet egyetlen lépésben akár több állomásra is elküldhető.

A hálózatra vonatkozó programozási előírásokat az IBM Network Technical Reference Manual tartalmazza. A kézikönyvhöz lemezen mellékelt mintaprogramok és listák segítségével a rendszer kezelése könnyen és gyorsan elsajátítható.

A hálózatvezérléshez 21 parancs áll rendelkezésre. Programozásuk egyszerű, és nagyfokú rugalmasságot biztosítanak a felhasználónak. A hálózatkezeléssel kapcsolatos összes többi feladatot: a puffer- és DMA vezérlést, a protokollvezérlést stb. a NETBIOS (hálózatos be-kiviteli rendszer) végzi el. Többek között lehetőség van arra is, hogy a PC az adatátviteli utasítás végrehajtása alatt a csatolóegységgel párhuzamosan más feldolgozást végezhessen.

Teljesítménynövelés

A vizsgálatok azt mutatták, hogy a széles sávú hálózat egyébként is jelentős adatátviteli teljesítménye a csomaghosszak növelésével, a csatoló pufferméretének növelésével, a műveletvégzésnek, a lemezkezelésnek és a hálózatvezérlésnek egymással párhuzamosra szervezésével tovább növelhető.

A token ring rendszerű helyi hálózat

Az IBM 1986-ban jelentette be második PC alapú helyi hálózatát, amely 4 Mbps sebességű, vezérelt jelzéses (token passing) hálózat-hozzáféréssel működő, alapsávú rendszerű hálózat. Az alkalmazott kábeltípustól függően 72, illetve 260 munkaállomást tud rendszerbe kapcsolni kétutas sodrott érpárú kábel (4 vezeték) segítségével.

A hálózat topológiája logikai gyűrű, fizikailag csillagelrendezés. A token ring hálózatot MS—DOS operációs rendszer működteti. Ettől a hálózattól és a benne alkalmazott megoldásoktól azt várják, hogy azokat más IBM-termékekhez hasonlóan az ipar „de facto" szabványként fogja elfogadni, és hogy a szoftverfejlesztők jelentős része erre a hálózatra készíti új alkalmazásait.

Hardverrészegységek:

— IBM Token Ring Network PC adapter (token ring hálózati csatoló),
— Calling System (kábelezési rendszer),
— IBM Token Ring Network Multistation Acces Unit (koncentrátor).

Szoftverrészegységek:

— IBM Token Ring — IBM PC Network Interconnect Program (IBM hálózatközi kommunikációs program),
— IBM Asynchron Communication Server Program (aszinkron adatátvitel-vezérlő program),
— IBM Token Ring Network PC Adapter Hardware Maintenance and Service (diagnosztikai program).

Hálózati csatolóegység

A csatolóegység segítségével IBM PC-k vagy velük kompatibilis személyi számítógépek kapcsolhatók össze számítógépes hálózattá. A csatoló egykártyás kivitelű, speciális, a Texas TMS 380 áramkörkészlet nagy integráltságú elemeiből épül fel. A különböző hálózatvezérlési, hiba-ellenőrzési és diagnosztizálási funkciókat beépített mikroprogramjai segítségével valósítja meg. A csatolóval megvalósított fizikai és adatkapcsolati funkciók kielégitik az IEEE 802 szabvány előírásait. A csatoló diagnosztikai programja forgalmazás közben ellenőrzi a csatoló működését és az adatutak (kábel) állapotát.

A kábelezési rendszer

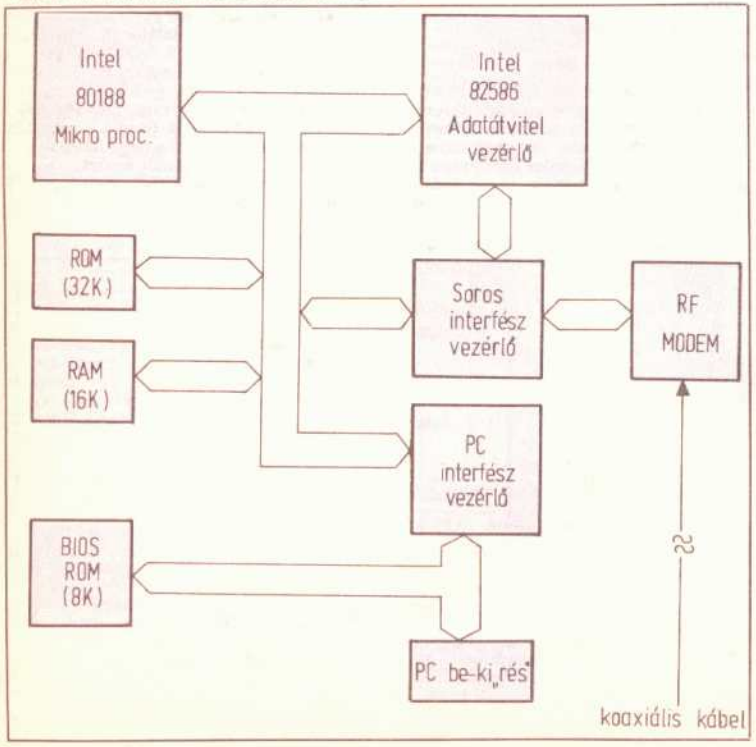
A token ring-hálózat logikai szinten gyűrű-, fizikai szinten csillagtopológiát valósít meg. A „kombinált" megoldás kialakítására azért volt szükség, hogy ezáltal a hálózat megbízhatóságát növeljék, azaz kiküszöböljék a gyűrűs hálózatoknak azt a hiányosságát, hogy a hálózat bármelyik munkaállomásának meghibásodása működésképtelenné teheti a teljes hálózatot. Ez azért fordulhat elő, mert a gyűrű topológiájú hálózatban a kábelre az egyes munkaállomások sorosan kapcsolódnak vonali adóvevőikkel, és ezáltal szakaszokra bontják a kábelt, ami végső soron csökkenti a rendszer megbízhatóságát a sín topológiájú hálózatokhoz képest.

A token ring-hálózat kétféle kábeltípussal működtethető: egy olcsóbb, négyeres, sodrott érpárú, árnyékolatlan kivitelű telefonkábellel, mellyel max. 72, jobbára egy épületen belül telepített munkaállomás kapcsolható rendszerbe, valamint egy drágább, kimondottan adatátviteli célokra készült, sodrott érpárú kábellel, mellyel max. 260 munkaállomás kapcsolható össze.

A token ring-hálózat adatútjai egyirányúak, azaz a hálózatban külön van az adás irányú és külön a vétel irányú adatút. Ezért van szükség a négyeres kábelre.

A kábelezési rendszer részegységei az

2. ábra. A helyi hálózati csatolóegység blokkvázlata



adatátviteli kábel, a csatlakozók, az elosztók, a fali szerelvények stb. A PC és a koncetrátor közötti kábelszakasz négyféle: 2,4 m, 9,1 m, 22,9 m és 45,7 m hosszúságban készül.

A koncentrátor

A token ring-hálózat koncentrátora passzív egység, amely a munkaállomásokat dugaszolással csatlakoztatja a kábelgyűrűhöz. A koncentrátor alkalmazásával létrehozott vegyes topológiával sikerült a gyűrűs hálózat már említett hiányosságát kiküszöbölni.

A megoldás azzal az előnnyel is jár, hogy ha egy gazdasági egység munkaállomásait ugyanazon koncentrátoron keresztül kapcsolják a hálózatra, annak forgalma csak kismértékben „terheli" a gyűrű forgalmát.

Egy koncentrátorral maximum 8 PC csatlakoztatható a kábelgyűrűre. Több kábelcsatlakozó összekapcsolásával nagyobb állomásszámú hálózatok is kialakíthatók.

A koncentrátor működési elvét, munkaállomás- és kábelgyűrű-kapcsolatát a *3. ábra* szemlélteti.

A koncentrátorban az egyes munkaállomásoknak a gyűrűre való rákapcsolását jelfogókkal valósítják meg. Ezek alaphelyzetükben, amikor nem kapcsolódik a munkaállomás a gyűrűre, zárt állapotukkal biztosítják a gyűrű ohmikus folytonosságát. Ha a munkaállomást kábeldugaszolással rákapcsolják a koncentrátorra, a zárt jelfogó bont és sorosan beiktatja a munkaállomást a gyűrűbe. Ezt úgy valósították meg, hogy minden csatolóegységben van egy jelfogómeghajtó áramkör, amely a munkaállomás koncentrátorra való csatlakoztatása esetén automatikusan meghúzatja „saját" jelfogóját, és ezáltal mintegy rákapcsolja önmagát a gyűrűre, illetve nem húzatja meg azt, ha a készülék hibás vagy nincs bekapcsolva. Ilyenkor a munkaállomás automatikusan lekapcsolódik a gyűrűről.

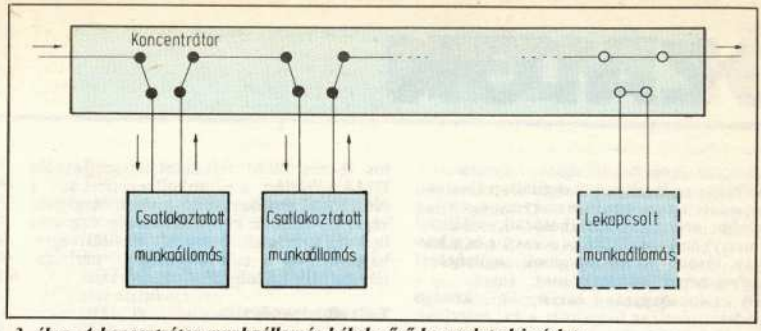
Az interfészek

A token ring-hálózatban kétféle hálózati interfész létezik: az Advanced Program to Program Communication (APPC), valamint a NETBIOS (Network Basic Input-Output System) interfész. A token ring hálózat a széles sávú hálózatban megvalósított alapszintű kommunikációs szolgáltatásoknál magasabb szintű szolgáltatásokat nyújt a felhasználónak. Ezek az APPC interfészen keresztül érhetők el, míg az alapszintű szolgáltatások mindkét interfészen keresztül elérhetők. A magas szintű szolgáltatások azt jelentik, hogy a hálózat különböző gépein futó tranzakció-kezelő alkalmazói programok adatkapcsolatot teremthetnek egymással.

A token ring hálózat munkaállomásai a hálózati erőforrásokat a NETBIOS alkalmazói programozási interfészen keresztül, konkurens módon érhetik el.

A hálózatban a NETBIOS interfészen keresztül kommunikáló alkalmazói programok mind a token ring-, mind a széles sávú hálózaton, az APPC/PC interfészen keresztül forgalmazó alkalmazások viszont csak a token ring-hálózaton futtathatók.

Azok az alkalmazói programok, melyek a széles sávú hálózat fájl- és nyomtatószerver szolgáltatásait használják, a token ring-hálózaton is igénybe vehetik ugyanezeket a szolgáltatásokat.



3. ábra. A koncentrátor-munkaállomás-kábelgyűrű kapcsolat elvi vázlata

IBM hálózatközi adatátvitel-vezérlő program

A program segítségével egy széles sávú és egy token ring típusú IBM helyi hálózat kapcsolható össze olyan céllal, hogy a különböző hálózatokhoz tartozó munkaállomások párbeszédes kapcsolatot létesíthessenek egymással. Maga a program a két hálózat számára közös IBM PC-n fut, a hálózatokkal való kapcsolat a PC-be dugaszolt két különböző hálózati csatolón keresztül valósul meg.

A hálózatközi adatátvitel-vezérlő program segítségével megvalósított adatkapcsolat feltétele, hogy az alkalmazói programok NETBIOS interfészen keresztül működjenek.

Aszinkron adatátvitel-vezérlő program

Ez a program bármely IBM helyi hálózathoz kapcsolódó, kommunikációs szerver funkciót is ellátó munkaállomáson háttérprogramként futtatható. A program egyszerre két aszinkron vonalat kezel, rajta keresztül további IBM PC-k kapcsolódhatnak rá a helyi hálózatra, és ezáltal mintegy a helyi hálózat külső munkaállomásaivá válnak. A kapcsolat a nyilvános távbeszélőhálózaton keresztül is kiépíthető.

Diagnosztikai program

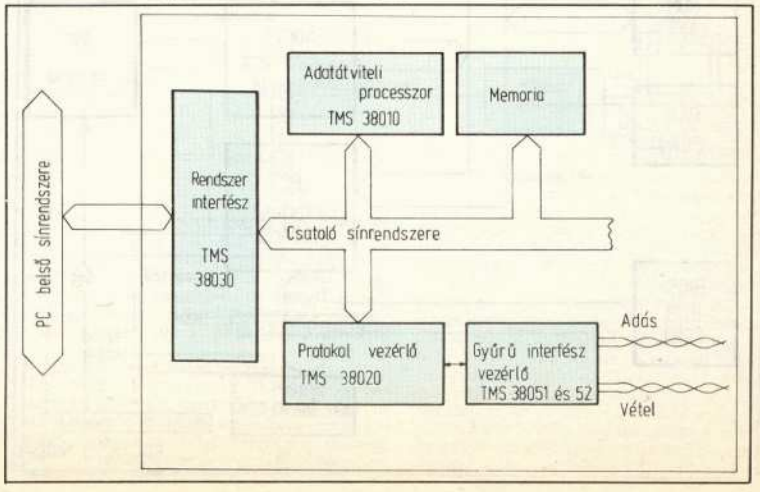
A hálózati csatolóegység és a hálózati gyűrű hibáinak pontosabb diagnosztizálását segíti elő ez, a csatolóegységben tárolt program. Az átvitel közbeni diagnosztizálást végző, ROM-okban tárolt programnál magasabb szintű hibabehatárolást tesz lehetővé.

A hálózati csatolóegység megvalósítása

Az IBM token ring-hálózat csatolóegységét a Texas Instruments TMS 380 elemkészletéből építették meg. Ez az elemkészlet a következő áramköri egységeket tartalmazza: TMS 38010 adatátviteli műveletvégző (processzor), TMS 38020 protokollvezérlő, TMS 38030 rendszerinterfész-vezérlő, TMS 38051 és 52 gyűrűinterfész-vezérlő. Az áramkörök a csatolóegység sínrendszerére csatlakoznak (lásd a *4. ábrát*). A csatolóegység kielégíti a helyi hálózatokra vonatkozó IEEE 802 előírásokat.

A rendszerinterfész-vezérlő 40 Mbps-os átviteli sebességgel forgalmaz a csatolóegy-

4. ábra. A token ring hálózati csatoló blokkvázlata



ség és a csatolt PC közötti rendszerinterfészen keresztül (DMA-átvitel). Az átvitel magas szintű parancsok segítségével hajtható végre.

A Texas token ring típusú helyi hálózat elemkészlete

A Texas Instruments az IBM felkérésére fejlesztette ki az IBM PC-ket token ring (vezérelt jelzéses) típusú, gyűrű elrendezésű, helyi hálózattá összekapcsoló speciális áramkörkészletét. Az áramkörkészlet típusjele: TMS 380. Az IBM helyi hálózat sodrott érpáron vagy optikai kábelen, 4 Mbps átviteli sebességgel működik.

A TMS 380 elemkészletből megépített helyi hálózati csatolóegy „tudja" az IBM PC-k és velük kompatibilis gépek helyi hálózatban való működtetéséhez szükséges összes funkciót, valamint a rendszer megbízhatóságát növelendő, a csatolóegység és az adatutak működését „menet közben" ellenőrző diagnosztikai funkciókat is.

Integrált architektúra

A csatolóegység felépítését tekintve, IBM PC-be — XT vagy AT — dugaszolható egykártyás kivitelű egység, amely a PC belső sínrendszere (rendszerinterfész), valamint a sodrott érpárú helyi hálózati kábel közé kapcsolódik *(1. ábra)*.

A TMS 380 áramkörkészlet az alábbi funkcionális egységekből áll:

Megnevezés	Típusjel	A tok lábszáma
Rendszerinterfész-vezérlő (DMA)	TMS 38030	100
Kommunikációs processzor és átmeneti tároló (RAM)	TMS 38010	48
Protokollvezérlő és vezérlőprogram-tároló (ROM)	TMS 38020	48
Kábelgyűrű-interfész adó-vevők	TMS 38051	22
Kábelgyűrű-interfészvezérlő	TMS 38052	20

Az áramkörkészlet NMOS technológiával készült nagy integráltságú tokokat (VLSI) tartalmaz, aminek legfőbb előnye a nagy teljesítményű (4 Mbps) adatátvitel, a kevés tokból megépített csatolóegység kis mérete, valamint a kis teljesítményfelvétel.

Az áramkörkészlet által megvalósított adatátviteli struktúra követi az IEEE 802 előírásait és az OSI nyílt architektúra hétréteges szerkezetét *(2. ábra)*. A struktúrából látható, hogy az alsó négy réteg a hálózat működését, a felső három réteg a hálózat használatát határozza meg.

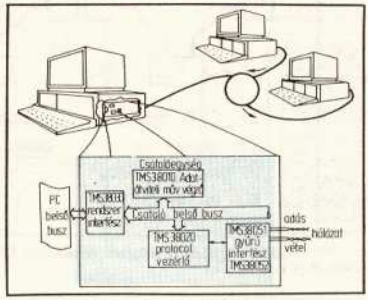
Az egyes rétegek és a hozzájuk tartozó funkciók az alábbiak:

— A fizikai réteg határozza meg a hálózat elektromos és fizikai csatlakoztatását.

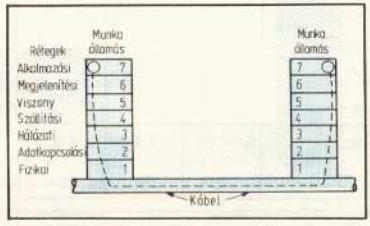
— Az adatkapcsolati réteg határozza meg a hálózat adatformátumát és a hálózathoz való hozzáférés módját (token passing = vezérelt jelzéses). Az adatkapcsolati réteg két alrétegre oszlik, a hálózatelérést, illetve a logikai kapcsolatot vezérlő alrétegekre. A funkcióknak az adatkapcsolati rétegeken belüli megoszlását és a szabványokkal való kapcsolatát a *3. ábra* szemlélteti.

— A hálózati réteg határozza meg az adatút felépítésének módját, illetve az adatok hálózatok közötti átirányítását (több helyi hálózat összekapcsolásával létrehozott számítógépes hálózati rendszerek).

Az ábra jól szemlélteti az IEEE 802 szabvány egyes alpontjainak kapcsolatát egyrészt az adatkapcsolati rétegszerkezet alrétegeivel, másrészt a különböző hálózatelérési eljárásokkal, illetve topológiákkal (hálózati elrendezés). Az ábrából látható, hogy például az IEEE 802.3 a sintopológiára és a CSMA/CD hálózatelérési eljárásra, a 802.4 a sintopológiára és a vezérelt jelzéses hálózatelérési eljárásra, a 802.5 pedig csillagban kapcsolt gyűrűtopológiára és a vezérelt jelzéses hálózatelérési eljárásra vonatkozik (ezt valósítja meg a TMS 380 elemkészlet is). A *szállítási réteg* gondoskodik az adatcsomagok „feladásáról" és ellenőrzi azoknak a célállomásra való sikeres „megérkezését" (nyugtázás). A *viszonyréteg* határozza meg a hálózatoldali felhasználói interfészt és vezérli az adatátvitel „párbeszédességét". A *megjelenítésréteg* határozza meg az alkalmazói program adatstruktúráját a viszonyréteg felé. Az *alkalmazási réteg* teszi hozzáférhetővé a hálózat szolgáltatásait az alkalmazói programok számára.



1. ábra. A TMS 380 elemkészlet



2. ábra. Az OSI referenciamodell

A hálózati topológia

A TMS 380 elemkészlet tehát egy ún. vegyes topológiájú (logikailag gyűrű, fizikailag csillag elrendezésű), vezérelt jelzéses eljárással működő hálózatot valósít meg. Ennek működését az IEEE 802.5 pontja rögzíti.

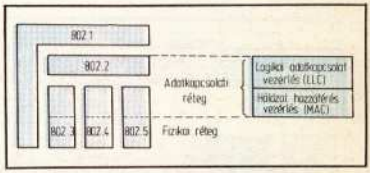
A vegyes topológiával a csillaghálózatok pont—pontközi kapcsolatában rejlő előnyöket sikerült átültetni a gyűrűs hálózatba, ami csökkenti a rendszer karbantartásigényét és növeli megbízhatóságát. A hálózat topológiája a *4. ábrán* látható.

A hálózati munkaállomások a hálózati csatolóegységen és egy nyúlványkábelen keresztül kapcsolódnak a hálózati gyűrűre. A kapcsolatot fizikailag egy passzív áramköri egység, a koncentrátor valósítja meg. Ennek elvi felépítését az *5. ábra* szemlélteti.

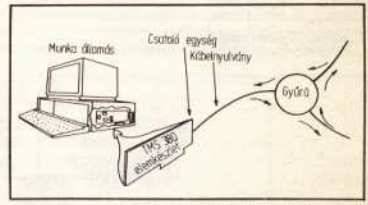
Egy koncentrátor maximum 8 jelfogót tartalmaz. A hálózatba több koncentrátor is bekapcsolható. Minden hálózati állomáshoz — legyen az akár munkaállomás, akár valamilyen szerver funkciót ellátó állomás — a koncentrátorban egy jelfogó tartozik. A jelfogók alapállapotukban, ha nincs bedugaszolva a nyúlványkábel a koncentrátorba, vagy nem ad ki jelfogó-meghúzatójelet az állomás csatolóegysége, kiiktatják az állomásokat a hálózatból; meghúzott állapotukban, ha az állomások nyúlványkábelei be vannak dugaszolva a koncentrátorba, illetve a csatolóegység üzemképes és kiad egy meghúzatójelet, rákapcsolják az állomásokat a hálózati gyűrűre. Ezzel a megoldással küszöbölték ki a tervezők, hogy az esetlegesen meghibásodó állomás üzemképtelenné tegye a hálózatot.

A vezérelt jelzéses hálózatelérési módszer

Az egyes állomások hálózaton belüli forgalmazási jogosultságát a hálózatelérési



3. ábra. Az IEEE 802 helyi hálózati szabvány



4. ábra. A csillagkábelezésű gyűrűs hálózati elrendezés

protokoll vezérli. Az angol megnevezésben (token passing) szereplő token olyan hálózat-hozzáférési jogot biztosító üzenetet jelent, amely a gyűrűben állomásról állomásra körbejár. A token passing protokoll jellemzője, hogy egy adott időpillanatban csak egyetlen hálózati állomás számára biztosít forgalmazási jogot, ezzel kerülve el a két vagy több állomás egyidejű forgalmazásából adódó üzenetütközést. A forgalmazási jog csak egy adott ideig tart, ezalatt forgalmazhat az állomás, ha akar. Ha nem, tovább kell adnia ezt a jogot — a tokent — a sorrendben őt követő állomásnak.

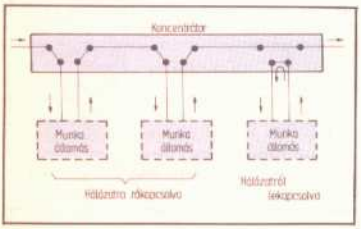
Ha a forgalmazni kívánó állomás megkapta a szabad tokent, foglaltra állítja azt, és belepakolja az előzőleg összeállított üzenetét. Az üzenet elküldése után az egyes állomások a gyűrűre való rákapcsolódásuk sorrendjében megvizsgálják azt: nekik címezték-e, és ha igen, beolvassák annak adatrészét csatolójuk átmeneti tárolójába, a vételi pufferbe, majd nyugtázzák a vételt

(pozitív nyugta = sikeres vétel, negatív nyugta = sikertelen vétel, azaz a forgalmazást meg kell ismételni) az üzenet megfelelő bitjeinek beállításával. Az így nyugtává módosított üzenet ezt követően tovább halad a hálózatban, az állomások felismerik és továbbengedik, végül a forgalmazó állomás a nyugta jellegétől függően vagy kiemeli a hálózatból és helyébe egy üres tokent helyez pozitív nyugta vétele esetén, vagy negatív nyugta vétele esetén megismétli a forgalmazást. Így érhető el, hogy egyszerre mindig csupán egyetlen üres token legyen a hálózatban. A megoldás nagy előnye, hogy a hálózatban a gépi válaszidők hossza garantált, ami különösen valós idejű rendszerekben való alkalmazás esetén előnyös.

A tokennek kétféle formátuma létezik: az üres formátumú token 3 bájt hosszú, az üzenet vagy keret formátumú token pedig 21 bájt + adatrész hosszúságú *(6. ábra)*. A token passing hálózati elérésű rendszer működésének elvi vázlatát a *7. ábra* szemlélteti.

A helyi hálózati vezérlés korlátai

Korábban főleg olyan helyi hálózati architektúrákat valósítottak meg, amelyek-



5. ábra. A koncentrátor

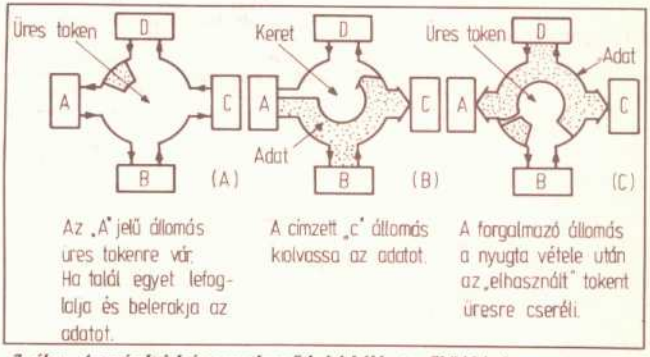
Az említett elvek szerint tervezett helyi hálózati csatoló az alábbi főbb funkcionális részegységet tartalmazza:

— általános célú mikroprocesszort,
— az adatok átmeneti tárolására szolgáló írható-olvasható típusú tárolót (RAM),
— hálózatvezérlési és diagnosztikai funkciókat ellátó programtárolót (ROM),
— a PC rendszerinterfészét és a helyi hálózati interfészt vezérlő áramköröket.

A ilyen csatoló blokkvázlatából — lásd a *8. ábrán* — könnyen belátható, hogy annak nem speciális, nagy integráltságú áramkörökből való megépítése elhelyezési nehézségeken túl további problémákat is felvet, nevezetesen a meglehetősen nagy disszipációt.

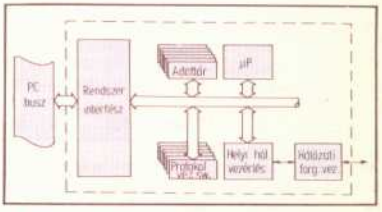
Az integrált architektúra előnyei

A TMS 380 áramkörcsalád NMOS, illetve bipoláris technológiával készült, és az előzőekben említett funkciókat összesen öt VLSI tokkal és néhány kiegészítő áramkörrel valósítja meg, egyetlen PC-be dugaszol-



7. ábra. A vezérelt jelzéses rendszerű helyi hálózat működési elve

6. ábra. Tokenformátumok



8. ábra. A hálózati csatoló blokkvázlata

Az elemkészlet funkcionális egységei (IC-tokok)

Rendszerinterfész-vezérlő

Típusjele: TMS 38030. Az interfészen megvalósított átviteli teljesítmény 40 Mbps. A csatolóegység 8, 16 és 32 bites iA PX 86, valamint a 32000 sorozatú mikroprocesszorokat, illetve a 16 és 32 bites 68000 sorozatú mikroprocesszorokat tartalmazó gépekhez csatlakoztatható. Az átvitel magas szintű utasításokkal (TRANSMIT, RECEIVE stb.) vezérelhető. A címtartomány 24 bites. DMA átvitellel, összefűzéssel nem folytonos címtartományú memóriaterületek is átvihetők. Lehetőség van „ömlesztett" átvételű vagy ciklusloposos üzemmódban való működtetésre is.

Az adatávitel-vezérlő processzor

Típusjele: TMS 38010. 16 bites műveletvégzőt és vele közös tokba épített 2,75 kbájt írható-olvasható tárat (RAM) tartalmaz.

ben a hálózatvezérléssel kapcsolatos összes funkciót a hálózati állomások központi műveletvégző egységei láttak el. Napjainkban már szinte kizárólag olyan rendszerek készülnek, melyek beépített intelligenciával (mikroprocesszor, tárolt programok stb.) rendelkeznek, és főleg a közvetlen hálózatvezérléssel kapcsolatos funkciók ellátásával nagymértékben tehermentesítik az egyes állomások műveletvégző egységeit.

Ez az irányzat különösen nehéz feladat elé állította a helyi hálózati csatolókártyatervezőket, mivel kis felületen sok IC-tokot kellett elhelyezniük. Ezen a területen hozott forradalmi változást a TMS 380 áramkörkészlet megjelenése.

ható kártyán. A csatolóegység blokkvázlata a *9. ábrán* látható.

Az elemkészlettel rugalmas DMA (Direct Memory Access = közvetlen memóriahozzáférésű) rendszerinterfészt valósítottak meg a PC belső sínrendszere felé, beleértve a dugaszolhatóságot is. A TMS 380 elemkészletből felépített csatolóegység alkalmazása több előnnyel jár: a hálózati állomásokhoz dugaszolásszintű csatlakoztathatóság valósult meg; a munkaállomásokat tehermentesítették a hálózatvezérléssel kapcsolatos funkciók ellátása alól, amitől nagymértékben megnőtt a helyi hálózatnak mint számítógépes rendszernek a teljesítménye; és igen kedvező az alacsony ár.

A tároló az üzenetek adás és vétel irányú átmeneti tárolására szolgál. A processzor működését a TMS 38020 típusjelű hálózatvezérlőben tárolt program vezérli.

A RAM tárkapacitás bővíthető, a maximális memóriabővítési lehetőség 42 kbájt. A memória teljes egészében paritásellenőrzött.

A hálózathoz férést vezérlő egység

Típusjele: TMS 38020. Feladata a helyi hálózati adatforgalom IEEE 802.5 szerint történő, 4 Mbps-os sebességű vezérlése. A processzor működését vezérlő ún. protokollvezérlő programot a tok 16 kbájtos ROM-jába égették be.

A protokollvezérlő program funkciói: a hálózati gyűrű forgalomvezérlése, ezen belül üzenetek címvizsgálata, üres token figyelése, üzenetek forgalmazása és fogadása, átmeneti tárvezérlés, a tokenprioritás ellenőrzése, valamint a hálózatdiagnosztikai funkciók.

A hálózati gyűrű és a csatolóegység átmeneti tárolója közötti nagy sebességű adatátvitelt két adási és két vételirányú DMA csatorna alkalmazásával valósították meg. Az adás-, illetve vételirányú adatátvitel CRC (ciklikus redundancia-ellenőrzés) ellenőrzött, a belső adatutak és a regiszterek paritás-ellenőrzöttek.

A hálózati gyűrű oldali interfészvezérlő

Típusjele: TMS 38051 és TMS 38052. Funkciói: a gyűrű interfészillesztése (adás- és vételirányban analóg-digitális jelátalakítás), a gyűrű forgalomvezérlése, a gyűrűórajel-előállítása, adatdetektálás, a koncentrátor jelfogómeghajtása, adatút-visszahurkolás (diagnosztikai funkció) és a kábel hibaellenőrzése.

Diagnosztika

A helyi hálózati csatolóegység működőképességét az alábbi hibaellenőrzések és vizsgálatok biztosítják:

— belső adatutak paritásellenőrzése,
— címellenőrzés,
— CRC áramkörök vizsgálata,
— gyűrűinterfészen belüli visszahurkolásos vizsgálat (elvi vázlatát lásd a *10. ábrán*),
— a kábelnyúlvány vizsgálata,
— a csatoló, a kábelnyúlvány és a koncentrátor együttes vizsgálata (elvi vázlatát lásd a *11. ábrán*).

A csatoló működését azokkal a diagnosztikai programokkal ellenőrzik, melyeket a protokollvezérlő olvasható tárolójában helyeztek el (ROM).

A hibafelfedő rendszer biztosítja, hogy a csatolóegységben, illetve a kábelnyúlványban előforduló bármilyen hiba esetén a hibás egység automatikusan lekapcsolódjon a kábelgyűrűről, így biztosítva a hálózat többi állomása számára a rendszer működésképességét.

A felfedett hiba automatikusan indítja egyrészt a hálózatról való lekapcsolódást, másrészt a diagnosztikai programot, amely ellenőrzi a részegységek működését. Ha nem talál hibát, a csatoló újból rákapcsolódik a gyűrűre. Ha azonban a diagnosztikai program észleli a hibát, akkor jelzi a hibaállapotot a munkaállomás képernyőjén.

Hibabehatárolás

A csatolóegység a rendszer javítható hibáit automatikusan küszöböli ki, ha tudja, miközben erről értesítést küld a hálózatfelügyelőnek. Kemény hibák esetén automatikusan elvégzi a diagnosztika a hibahely behatárolását, de mivel ennek kiküszöbölése már kezelői beavatkozást igényel, lekapcsolódik a hálózati gyűrűről, és jelzést küld a szóban forgó állomás, illetve a hálózatfelügyelői készülék képernyőjére.

A hálózatos szolgáltatások szabványosítása tette lehetővé, hogy a hálózatos alkalmazásokat fejlesztők rendszereiket a hardvergyártóktól függetlenül fejleszthessék. Mivel ezeket a hálózatos szolgáltatásokat magában a csatolóegységben valósítják meg, a felhasználók meglevő alkalmazói rendszereiket erre alapozva bővíthetik.

A TMS 380 elemkészlet a hálózatos szolgáltatásokat speciális utasítások — ún. MAC keretek — segítségével hajtja végre. A MAC kereteket az IEEE 802.5 definiálja. A TMS 380 rendszer a helyi hálózati gyűrű forgalomvezérlését 27 MAC keret segítségével valósítja meg. Ezek segítségével gyűjti össze például a hálózati munkaállomások csatolóegységeiben keletkező hibajelzéseket, figyeli és szükség szerint módosítja a helyi hálózat konfigurációját. Ugyancsak MAC keretek segítségével paraméterezhetők át a hálózat egyes egységei is.

Teljesítmény

A helyi hálózat gépi válaszidőben mért teljesítményét döntően négy tényező határozza meg: az adatátviteli sebesség, a hálózatelérési protokoll által „okozott" késleltetés, a helyi hálózat és a munkaállomás közötti adatátvitel ideje és a munkaállomás protokollvezérlő szoftverének hatékonysága.

A vezérelt jelzés protokoll

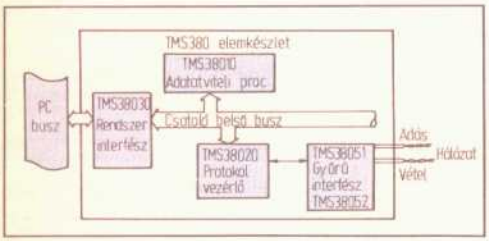
A protokoll biztosítja, hogy a rendszer gépi válaszidői alig függnek a hálózati terhelés és a forgalom változásától, ami jelentősen eltér az ütköztetés rendszerű protokolloktól (például CSMA). A TMS 380 elemkészletnek köszönhető az is, hogy a rendszerben az egyes hálózati állomások csak minimális, állomásonként kb. 2,5 bit tokenkésleltetést okoznak a gyűrűn.

A munkaállomás és a gyűrű közötti adattovábbítás

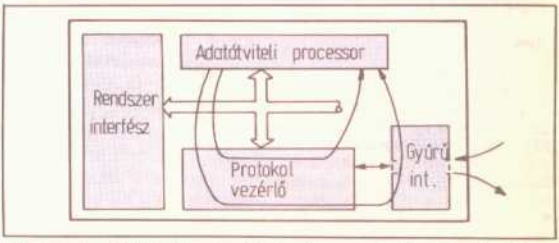
A munkaállomás és a hálózat között a csatoló 4-4 Mbps-os sebességgel teljes dup-lex átvitelt valósít meg mind adási, mind vételi irányban. A csatolóegység belső sínjének átviteli sebessége 48 Mbps (csatolón belüli átvitel), illetve 40 Mbps a csatoló belső sínje és a munkaállomány perifériainterfésze között (DMA átvitel).

A munkaállomás-protokollvezérlő szoftver hatékonysága

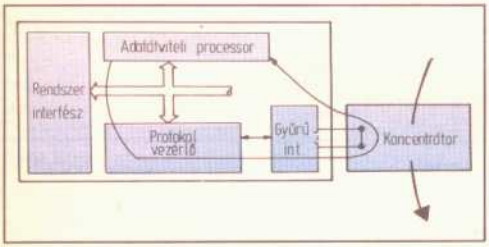
A protokollvezérlés hatékonysága nagymértékben növelhető a TMS 380 elemkészlet alkalmazásával.



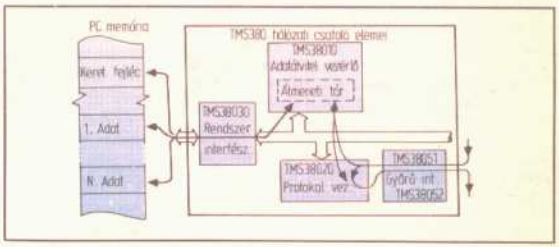
9. ábra. Az integrált architektúra



11. ábra. A gyűrűinterfész vizsgálata a koncentrátorban visszahurkolva



10. ábra. A gyűrűinterfész vizsgálata belső visszahurkolással



12. ábra. A csatoló adatútjai

A csatolóegység az alábbi üzemmódokban működtethető:
— ciklusloposos,
— programozott „ömlesztett" átvitel,
— magas szintű parancsokkal vezérelhető „összefűzött lista",
— lekérdező, illetve megszakításos rendszerű működésmód,
— listafelfüggesztéses,
— nem folyamatos címterületen tárolt adatblokkos DMA átvitel.

A munkaállomás és a hálózat közötti adatátvitel csatolóegységen belüli útját a *12. ábra* szemlélteti.

A rendszerinterfész paraméterezése

A munkaállomásnak a hálózatra történő rákapcsolása előtt kell elvégezni a csatolóegység interfészének a munkaállomás-rendszerinterfészhez való illesztését. Maga az illesztés a munkaállomásról hajtható végre. Ennek keretében kell megadni egyes paramétereket: megszakítási állapotok, vételi csatornák, tárbővítés, címbeállítás stb. A munkaállomás és a csatolóegység közötti átvitelt a rendszer parancs- és állapotblokkok segítségével hajtja végre. Ilyen blokkok a rendszerparancsblokk = System Command Block (SCB) és a rendszerállapotblokk = System Status Block (SSB).

Keretátvitel

Adásirányú átvitel

Keretek adásirányú átvitele a következőképpen hajtódik végre. A munkaállomás forgalmazási szándékát jelezve megszakítást kér, majd egy SCB-t küld a csatolóegység felé. Az SCB tartalmazza a forgalmazási parancskódját és az átviendő adatblokk munkaállomási főtárbeli kezdőcímét. Ezt követi maga a tényleges átvitel, melynek során a csatoló DMA-val beolvassa saját átmeneti tárolójába az átviteli listát, majd ebből összeállítja az átvinni kívánt adatokat tartalmazó keretet. A csatoló ezután kivárja az első üres tokent, és benne elküldi az előzőleg összeállított keretet, amely állomásról állomásra körbejár a gyűrűben, míg a címzett állomáshoz érve, az beolvassa az adatrészt és nyugtázza a sikeres vétel tényét a keretet küldő állomás felé. Ha ez a forgalmazó állomás „visszakapta" a hálózatból a nyugtakeretet, sikeres vétel esetén kiemeli azt onnan, és helyére egy üres keretet helyez, amit azonnal továbbad a sorrendben őt követő állomásnak forgalmazásra. Ezután a csatoló aktualizálja az SSB-t, majd továbbítja munkaállomása felé.

Vételirányú átvitel

A hálózatból érkező keretek fogadása a forgalmazáshoz hasonlóan történik. Ilyenkor az SCB a hálózatból érkező adatblokk munkaállomásbeli tárkezdőcímét tartalmazza. A keret vételét követően a csatoló lekérdezi a munkaállomásától a tárolás kezdőcímét (vételi lista), és ennek megfelelően tölti be a vett és előtárolt adatblokkot a megadott főtárcímre, végül aktualizálja az SSB-t.

A TMS 380 elemkészletet az IBM token ring hálózat számára készítették, kereskedelmi beszerezhetőségéről nincs tudomásunk. A vele kapcsolatos információkat a Taxas a szocialista országok irányában jelenleg érvényben levő COCOM előírásoknak megfelelően kezeli.

CSEH KÁLMÁN

Számítógépes adatbankok mindenkinek

A személyi számítógépek elterjedése, használatuk mindennapivá válása megteremtette a maga sajátos, elektronikus világát. A korábban csak nagyobb cégek számára elérhető számítógépes adathálózatok az egyszerű nyugati polgár számára is hozzáférhetővé váltak. Midennapos lett az információ áramlása városok és kontinensek között az adathálózatokon és a postai telefonvonalakon keresztül — és ebből a PC-k sem maradhattak ki.

A nagy adatközpontok mellett először a számítógépes amatőrök köreiben, majd a hivatásos felhasználóknál is — mint ahogy a CB-rádió vált fokozatosan munkaeszközzé is — megjelentek a kisszámítógépekre vagy PC-kre épített automatikus miniadatbankok, elektronikus levelesládák. Kezdetben szerepük is a CB-vel volt rokon: az emberek egymás közötti kommunikációját, az egyén, a kisvállalkozók, vállalatok egymás közötti kényelmes és gyors kapcsolatát biztosították.

Innen már csak egy lépés kellett ahhoz, hogy egyes cégek belássák: az így kialakult rendszereket érdemes közönségszolgálati információk, termékkatalógus közzétételére is alkalmazni, vagy éppen ezzel, viszonylag kis költséggel, új fizető szolgáltatást bevezetni. És ha egyes cégek termékinformációs rendszert tartanak fenn, akkor érdemes azt rendelési lehetőséggel is kiegészíteni. Így ismerkedett még a világ a CASH (Computer Aided Shopping = számítógép segítette bevásárlás) fogalmával. A felhasználók továbbra is kihasználják az e rendszerek nyújtotta előnyt: a számítógépek segítségével levelezhetnek egymással, cserélhetik programjaikat, és írásos formában megvitathatják közös hobbijukat. Az egyik USA-beli fizető számítógépes hálózatban CB-szimulátorprogram is működik, melynek segítségével jelenleg 40 csatornán folyhat a információcsere, és azonos csatornán belül mindenki mindent hall, lát a PC képernyőjén, és bele is tud szólni, azaz írni.

Természetesen a folyóiratok is éltek ezzel a lehetőséggel, mert így olvasóikkal nemcsak közvetlen kapcsolatot tudnak tartani, hanem friss hírekkel is elláthatják őket. Az egyik legismertebb ilyen ingyenes rendszert az NSZK-ban a Microcomputer Zeitschrift c. folyóirat kiadója tartja fenn. A TEDAS — így nevezik e rendszert — adatbankul is szolgál: sok egyéb hasznos információ mellett a lapban megjelent egyes cikkek visszakereséséhez ad segítséget.

Ezeket, a Nyugat-Európában és Észak-Amerikában egyre mindennapibbá váló rendszereket CBBS (Computer Bulletin Board System) vagy még rövidebben BBS rendszereknek nevezik. Mivel a nagy nyilvánosságnak is szólnak, a könnyű elérhetőség kedvéért általában a nyilvános telefonhálózatot használják fel az otthoni PC és a rendszer közötti kommunikációra. Sőt némelyik BBS több adatátviteli hálózaton keresztül is elérhető. Nyugaton egyes cégek azzal is megkönnyítik e rendszerek használatát, hogy még a hívó telefonköltségét is magukra vállalják.

Igaz, hogy a hazai felhasználók szempontjából csak az ingyenes rendszerek jöhetnek számításba, de mivel ilyen is sok van, érdemes alaposabban foglalkozni a BBS-ekkel. E rendszerek jó része telefonon hazánkból is felhívható, bár rendszeres használatuk igencsak megnöveli a telefon-

1. ábra. Fizető BBS új felhasználómenüje, alatta a rendszerben már szereplő felhasználó számára kiadott menüoldal

User Validation will be ready within 24 hours for LLUMC Staff & Physicians.
Thank you. You have less than three Minutes to finish the Card.
– Medical Library

Checking correspondence
No correspondence for you at present.

LOMA LINDA UNIVERSITY MEDICAL CENTER
Online Medical Library
=== MAIN MENU ===

F ill out Library Card
A lter Terminal Display
E xit from the Library

Command:

YOU HAVE LOGGED – ON
A PRIVATE SYSTEM

Welcome to the LLUMC Medical Library!
A TBBS at 300/1200

I am Paul, your Medical Librarian

Normal Hours are: 5 p.m. - 7 a.m.(Mon-Thru)
and 24 hours Fridays 3 p.m.
until the following Monday at 7 a.m.

System up daytimes
when not in use by Library

Call voice ext.6260 other times
or Beeper #4082

First Name?

01/10/85 13:49:39
0000 BAUD CALL FROM: PAUL LLUMC MEDICAL LIBRARY
Type P to Pause, S to Stop listing

```
Hello, I am your Reference
Medical Librarian.
How may I help you?

I  nter-Library Loan Request Form.
M  edline Request Form.
C  all Online for the Librarian
T  houghts from the Library
E  nough of this.

Command:
```

```
LOMA LINDA UNIVERSITY MEDICAL CENTER
Online Medical Library
===        MAIN   MENU              ===

H  ousekeeping on the Board

B  rowse the Library
L  ibrary Computer Files      M  emos, Private
R  eference Desk              N  ew User Library Card
P  atron List                 C  onference Room
A  lter Terminal Display      E  xit from the Library

V  iew All the Boards
H  ousekeeping on the Board
```

```
Medical Library's
Conference Room

For Public Messages

P  hysicians Only—Private Conference Room
R  ead the Public Discussions
C  heck Conversation Topics
W  rite your Public Comments

E  nough of Messages

Command:
```

```
Physicians Only—a Private Conference
area for LLUMC & Related Facility
Medical Professionals

R  ead the Comments of your Colleagues
C  heck currently Discussed Topics
W  rite your Comments to your Colleagues

E  nough of this

Command:
```

```
Library News Files

M  edical Staff Alert
P  apers and Misc. Information
N  ews * Bytes
H  am Radio News
@  Medical Library Manual, part 1
!  Medical Library Manual, part 2
E  xit Browsing
```

2. ábra. Tipusos menük a Loma Linda University BBS rendszerében

számlát. A telefonvonalon történő kommunikáció akár 1200 baud sebességgel is képes elfogadható kapcsolatot létrehozni. (1 baud azt jelenti, hogy 1 másodperc alatt 1 bitnyi információt viszünk át. Baud, e mértékegység névadója, dolgozta ki a rádiógéptávíró adásoknál mai napig használt Baud-távírókódot.)

Ahhoz, hogy a számítógép igen-nem jeleit a hang továbbítására alkalmas telefonvonalon továbbítani lehessen, át kell alakítani hangjelekké. Ezt a feladatot a modem (modulátor—demodulátor) látja el. A postai szabványok határozzák meg, milyen frekvencia felelhet meg az 1 és melyik a 0 állapotnak. Az átvitel sebességét is szabványosították. A leggyakoribb a 300 baudos átvitel, de emellett előfordul az 1200 baudos, valamint a CEPT és a Prestel rendszerekben az 1200 baudos küldő- és 75 baudos adósebesség. Jelölése 1200/75. E rendszerek nem ASCII, hanem speciális adatátviteli kóddal dolgoznak. Használatukhoz külön dekóder kell. Ha jó a telefonhálózat, akkor az 1200 baud az a sebesség, amelyet éppen megfelelően visz át, és a leginkább elterjedt C64-es gép maximális adatátviteli sebessége is éppen ennyi. Ennél nagyobb átviteli sebességek a gyakorlatban csak adatátviteli hálózatokon és gép-gép közötti kommunikációs kapcsolatokban fordulnak elő.

A BBS-ek és a felhasználói PC hálózatok általában az ún. ASCII (American Standard Code for Information Interchange) kódot használják világszerte. Itt az egyes betűket, számokat, jeleket decimális számokkal jelölik. A 0—32-ig terjedő tartományban az utasítások, 33—47-ig a jelek, 48—57-ig a számok, majd 58—64 között ismét a jelek helyezkednek el. Ezután 65—90-ig a nagybetűs ábécé, majd néhány jel után, 97—122-ig a kisbetűs ábécé, és végül ismét jelek és utasítások következnek. Az ASCII kódot később 255-ig kibővítették. Ez a pótlás főleg különleges betűket és félgrafikus karaktereket tartalmaz, és gépenként, hálózatonként eltér.

A PC-ken az ASCII kód egyszerű utasítással hívható, és az adatátviteli sebesség is beállítható. De itt problémát jelent, hogy a továbbított adatformátumra is több egymás mellett élő, de nem kompatibilis szabvány létezik. Igaz, vannak olyan BBS-ek, amelyek alkalmazkodnak a mi adásmódunkhoz. Ezt egy-két karakter, majd a CR billentyű lenyomása után határozza meg a rendszer. Legtöbbször azonban a számítógépünkben futó emulátorprogram megfelelő megválasztásával nekünk kell ehhez alkalmazkodni. Általában hétbites szót használnak egy stopbittel, egyenlő paritással, de lehet nyolcbites szó, paritásbit nélkül, egy stopbittel. Ezeket az adatokat ugyanonnan szerezhetjük be, ahonnan megtudhatjuk a BBS telefonszámát: az illető szakfolyóirat impresszumában, összefoglaló táblázatban, vagy egy hirdetésben olvashatjuk.

3. ábra. HAM-rádió indítómenüje a Loma Linda University BBS rendszerében

```
ATTENTION!
HAM  RADIO  OPERATORS
New Ham Section In  B  rowse
Courtesy of two local boards
. . . . . .

Loma Linda U. Medical Center
Employees & Physicians
The Entire MS-DOS Public Domain
Library is available at $.50 ea.
Inland Empire Computer Group
===    Watch for Meeting Time    ===
CP/M and Apple CP/M also avail.

Checking correspondence
No correspondence for you at present.

Command:
```

```
C-64 Bulletin Board System
--------------------------
SYSOP: Mike Dunton

Log In At 1026h
Your First Name? Asher
Your Last Name? Fitzgerald

Standby...

VERY WELL ASHER FITZGERALD,
Is This Your First Time On The System?
yes

City/State? (30 chrs max) Stover, MO

Please supply a 6 character USER CODE
>AUSTIN
Name: ASHER FITZGERALD
City: STOVER,MO
Code: AUSTIN

Is the Information Correct? yes

Standby, Logging You In...

00:04:10 (Time Elapsed) Command > 0
Type 'S' to stop the listing.
Once stopped, 'S' will restart,
'A' will abort.

Msg #  : 564 - Ref 6543
From   : BOB BURKE
To     : JIM HANSEN
Subject: New C-64 game programs

Msg #  : 563 - Ref 6542
From   : BOB CRAMER
To     : SUE MARRIOTT
Subject: Help with assembly language

Msg #  : 562 - Ref 6541
From   : PAT DAVIS
To     : MARVIN VOGT
Subject: Political elections
```

4. ábra. Első bejelentkezés egy nem fizető USA BBS rendszerbe, és a rendszerben lévő üzenetek tartalomjegyzéke

A BBS és a hívó számítógép közötti kapcsolatot a modemek biztosítják. A hívó oldalán ezek általában akusztikus modemek, melyek a telefonkagyló hallgatóját és mikrofonját használják fel. Ezek annál is inkább elterjedtek, mert egyes postaigazgatóságok a vonalra történő elektromos csatlakoztatást büntetik, illetve engedélyhez kötik. Így tesz a Magyar Posta is, azzal a súlyosbítással, hogy elvben a telefon adatátvitelre történő használatát is engedélyeztetni kell a Posta Központi Távíró Hivatalnál. Ez utóbbi rendelkezés a gyakorlatban csak részben tartható be, mert nagyon sokszor voltam magam is tanúja annak, hogy a ha-

5. ábra. Ismételt bejelentkezés egy klub BBS-be, majd program áttöltése abból a saját gépbe. A rendszer a fájl áttöltésének befejeződését a "FILE TRANSFER COMPLETE" rendszerüzenettel jelzi

zánkban élő nyugati üzletember személyi számítógépe és a szállodai telefon segítségével lépett kapcsolatba cégével.

Egyes BBS rendszerek esetében ún. ringback system üzemmódban, kétszer kell a kapcsolat létrehozásánál felhívni az állomást. Amikor először hívjuk, úgy halljuk, mintha kicsöngés után valaki felvenné a hallgatót, esetleg egy magnó arra kér, hívjuk újra. Ekkor kapcsolódik be a számítógép. Három-öt percen belüli, másodszori hívásnál már a modem sípoló hangját halljuk: a rendszer kapcsolatra készen áll.

A BBS-ek bázisát általában valamilyen nagyszámítógép nyilvános fájlja vagy valamilyen „komolyabb" PC képezi, de lehet bő utalást találni a szakirodalomban C64 alapúakra is. A BBS rendszer működését speciális programcsomag és modem biztosítja az adatbank oldaláról. A hívónak elegendő egy modem és a megfelelő emulátorprogrammal ellátott személyi számítógép használata.

A BBS-ek felhasználás szerint lehetnek nyiltak, ez esetben bárki felhívhatja, használhatja őket. Ezek azok a rendszerek, amelyek számba jöhetnek a hazai PC-amatőrök számára is. A másik, és sajnos egyre terebélyesedő csoportjuk, a fizető BBS rendszerek, amelyek szolgáltatásai inkább hasonlítanak a nagy adatbankokéhoz. A használatukhoz szükséges jelszót (password) fizetés ellenében lehet megkapni az üzemeltetőtől, és az általa nyújtott szolgáltatásokért is fizetni kell. E rendszerek harmadik típusánál általában egy nagy rendszer valamely nyilvános részével állunk szemben. Használatához szintén jelszó szükséges, de ezt a BBS adataival együtt nyilvánosságra hozzák. Elég gyakori az ún. típusjelszó használata: például angol rendszereknél a „public", amerikaiaknál a „guest", vagy az IBM-kluboknál az „IBMPC" alkalmazása. Ettől akkor térnek el, ha a felhasználói kört szűkíteni akarják.

Egyes nyugati fizető rendszerek érdekessége, hogy hitelkártyát is elfogadnak. Ilyenkor a hívó az első bejelentkezéskor közli hitelkártyája számát, majd miután a rendszer ellenőrizte hitelképességét, néhány perccel későbbi, második bejelentkezésekor megkapja az érdemi használathoz szükséges személyes kulcsszót.

A BBS rendszerekben általánosan használt jelszók

ID	Jelszó
system	master
visitor	visitor
guest	guest
service	service
—	publik
INF	rendszernév
	IBMPC (IBM PC kluboknál)
USA vállalati információs fájlok esetében	vállalatnév (betűszó)

A BBS-ek általánosan használt utasításai

List	Tartalomjegyzék, illetve a lehetséges programok listája.
Load	Program betöltése a BBS-ből a saját gépbe.
Save	Program áttöltése saját gépből a BBS-be.
A	Az érdeklődőnek szóló összes üzenet lekérése (amennyiben a BBS-en keresztül, elektronikus postafiókként használva levelez másokkal).
E	Üzenet elküldése a BBS másik használójának.
O	A BBS-ben lévő üzenetek áttekintése.
P	Az utoljára olvasott üzenetet kérjük ismételni.
S	Az összes üzenet téma szerinti tartalomjegyzéke.
Help vagy H	Bővebb szöveges magyarázatkérés a rendszer sajátosságairól általában, illetve egy adott kérdés után arra a szituációra.
Bye vagy logoff	Kilépés a rendszerből.
SYSOP	A rendszer üzembentartója.
CTRL S	Az éppen folyó átvitel leáll.
CTRL Q	A leállítástól folytatja az adatátvitelt.
CTRL X	Visszatérés a rendszer fő menüjéhez (csak CTRL S után adható ki).

A CTRL billentyűt a Commodorokon a C-billentyű helyettesíti!

Az *1., 2* és *3. ábra* egy ilyen fizető rendszer protokollját és néhány menüjét mutatja be. Mint ezekből is látható, a BBS-ek utasításkészlete, különösen a közönségszolgálati és klubrendszereké, „barátságos". Arra késztették őket, hogy számítógépes alapismeretekkel nem rendelkező felhasználók is képesek legyenek kommunikálni velük. Innen ered a BBS-ek azon sajátossága is, hogy általában a működtető ország nyelvén küldik rendszerüzeneteiket. Utasításkészletük is a könnyebb használatot célozza, ezért sok esetben eltér a kialakult számítógépes gyakorlattól, sőt attól is függhet, milyen programrendszer fut a gépen. Ezért e rendszerek tervezői és készítői, gondolván e káoszra, legalább a HELP utasítást meghagyták egységesnek. Segítségével bő szöveges információt kaphatunk egy-egy rendszer sajátosságairól, speciális utasításkészletéről.

A klubrendszerek gyakran még olyan szolgáltatást is nyújtanak, hogy programot áttöltenek saját személyi számítógépünkbe vagy saját programunkat tudjuk így másoknak elküldeni. A bejelentkezés általában ezekbe a többnyire ingyenes rendszerekbe szintén meghatározott protokoll szerint történik. Első kapcsolatunk alkalmával a rendszer megkérdezi a címünket, néhány adatunkat, majd megkapjuk azt a jelszót, amellyel a későbbiekben e műveletet egyszerűsíthetjük. (Ugyanolyan illetlenség e kérdésekre hamis adatokkal válaszolni, mint névtelen levélben fenyegetni valakit.) Egy ilyen példát láthatunk a *4.* és az *5. ábrán*, egy C64 klub BBS rendszerének protokolljaként.

A BBS rendszerek a számítástechnika mindennapokba való bevonulását jelzi. És alighogy általánossá váltak a fejlett országokban, megjelent két testvérük is. Az egyik a HAM-rádió, amely tulajdonképpen CB-n keresztül történő adatátvitel, a másik a felhasználók számára a rádióamatőri hullámsávokon keresztül elérhető műholdas adatátvitel, amely szintén közelebb hozza egymáshoz a közös érdeklődésű embereket.

Hazánkban a nyugati BBS-ek felhasználása például szakirodalmi információk szerzéséhez, piackutatáshoz vagy egyszerűen emberi kapcsolatok kialakításához, most van kialakulóban. Hazai elterjedésük csak akkor várható, ha már általánosan ismertek lesznek, és alkalmazásuk olyan gyakorlattá vált, mint az, hogy az ember felhívja telefonon külföldi barátait. A HAM-rádióval megindultak az első kutatások a BME Központi Rádióklubjában, dr. Gschwind András vezetésével, és e cikk írója is megtehette a telefonon elérhető BBS-ek hasznosítása felé vezető út első lépéseit a KSH—SZÜV Budapesti Számítóközpontjában, dr. Majtényi György igazgató és lelkes munkatársainak támogatásával.

KIS JÁNOS

Fiatalok! Figyelem!

Rejtvénypályázatunk utolsó fordulójához értünk.

3. feladat

Készíts programot egy digitális óra szimulálására!

A követelmények:

— A program indításakor a hónap, a nap, az óra, a perc és a másodperc megadható legyen (célszerű megoldás: hónap, nap egyetlen négykarakteres sztring, óra, perc, másodperc egyetlen hatkarakteres sztring).

— A megjelenítés a képernyő bal felső sarkában történjen, az egyes adatok között egy betűköz kihagyásával (a számok előtt álló értéktelen nulla jegyek kiírása nem szükséges).

— A február mindig 28 napos.

— Egy tetszőleges billentyű lenyomására az alaphelyzetben óraként működő program váltson át a dátum kijelzésére, ismételt billentyűzés hatására térjen vissza óraüzemmódba stb.

A program értékelésénél a tárterület-lefoglalás is szempont: a program max. 500 bájt helyet foglalhat el a tárban. Törekedj rövid programra!

A beküldési határidő: 1986. december 20.

A megoldásokat írásban kérjük szerkesztőségünk címére:

Mikroszámítógép Magazin Szerkesztősége,
Budapest, Fő u. 68. IV. em. 452. 1027.

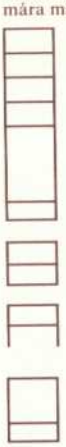
A borítékra írjátok rá: DIGITÁL rejtvénypályázat.

A helyes megfejtéseket és a nyertes nevét a jövő év elején közöljük.

A szerkesztőség

List — Microsoft BASIC-nél

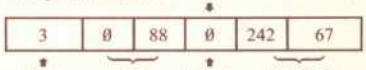
Pontosabban: list-bolondítás. A sokak számára már unalmas programsorok tárolása:



Mi van akkor, ha egy mutató nem a következő, hanem például az előző sorra mutat? A kérdés izgatott, kipróbáltam. Az eredmény végtelen listázási ciklus lett. Primón és HT-n már megcsináltam. Íme egy hordozható példa Primóra, HT-ra, Commodore-ra:

```
10 X$= ""
20 PO=PEEK(VARPTR(X$)+1)
      +256*PEEK(VARPTR
      (X$)+2)
30 H= INT((PO-8)/256):POKE PO+2
      PO-8-256*H,H:LIST
```

A vaptr ide mutat



típus ($) — név (X$) — hossz — tárolási cím

SZABÓ LÁSZLÓ

Adatbevitel-egyszerűsítő

Sok programozónak gondot okoz az adatok kényelmetlen bevitele az INPUT BASIC utasítás hiányosságai miatt. Ezért írtam ezt a programot, amely szubrutinként bármely programban felhasználható, és minden adatbevitelkor meghívható.

A meghívás előtt az O változóban kell megadni a beviteli mező oszlopszámát, az S változóban a sorszámát, a HO változóban pedig a kért adat hosszát. A szubrutin meghívása után a kért adatot az IN$ változóban kapjuk meg. A kurzor helyetti karakter az eredeti pozíciónál nem megy jobban vissza.

COMMODORE 64

A programban felhasznált változók:

O	—	oszlopszám
S	—	sorszám
HO	—	a beviteli mező maximális hossza
K	—	számláló
IN$	—	a bevitt adat
X$	—	az éppen bevitt adat
FI	—	oszlopfigyelő
j	—	ciklusváltozó
AS	—	a lenyomott billentyű ASCII kódja

A tényleges rutin a 20 000-es sortól kezdődik.

HEGEDŰS GÁBOR

```
7 DIM IN$(15):REM AZ ADATOK MAXIMALIS HOSSZA
10 REM BEMUTATO 10
20 S=10:O=28:HO=6:PRINT"■■■■■■■■■■■■■■■■■■■■■■REM A MAI DATUMOT ? "
25 S=10:O=28:HO=6:PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■EHHNN"
30 GOSUB20000:DA$=IN$:IF LEN(IN$)<>6 THEN 20
40 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■ JO A  DATUM ? (I/N)"
50 GET T$:IF T$="" THEN 50
60 IF T$="I" THEN 90
70 IF T$="N" THEN 20
80 GOTO 50
90 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■ A MAI DATUM : ";DA$
100 END
110 :
120 :
130 :
140 :
19000 REM ***** INPUT RUTIN *****
19010 :
20000 K=1:IN$="":X$="":FI=0:FOR J=1 TO 15:IN$(J)="":NEXT
20010 POKE214,S:POKE211,O:SYS58732:PRINTCHR$(122);:REM A KURZOR HELYETTI KARAKTER
20020 GETX$:IFX$=""THEN20020
20023 AS=ASC(X$)
20024 IF AS=20 AND O=FI THEN O=O+1:K=K+1
20025 IF AS=20 THEN K=K-1:POKE214,S:POKE211,O:SYS58732:PRINT" ";:O=O-1:GOTO 20010
20030 POKE214,S:POKE211,O:SYS58732:PRINTX$;
20040 IFAS=13THENPOKE214,S:POKE211,O:SYS58732:PRINT" ";:GOSUB21000:RETURN
20050 IN$(K)=X$:K=K+1
20055 IF K=HO+1 THEN :GOSUB21000:RETURN
20060 X$="":O=O+1:GOTO20010
21000 FOR J=1 TO K-1:IN$=IN$+IN$(J):NEXT:RETURN:REM ***** OSSZEGZO RUTIN *****
```

COMMODORE 64

Egy menüprogram

A menütechnikát alkalmazó programoknál találkozunk a program egymás utáni betöltéséből adódó problémákkal. Kétféle eljárást szoktak alkalmazni: a billentyűzetpufferbe történő beírást, vagy a behivandó program első sorában állítják be a BASIC-program vége mutatót — POKE45, PEEK(174): POKE46, PEEK(175):CLR. Az első módszer hátránya, hogy a képernyőformátumot elrontja, a másodikat pedig csak akkor alkalmazhatjuk, ha már pontosan tudjuk, mely programokat akarjuk a menüvel hívni.

Erre ad nagyon egyszerű megoldást a LOAD+RUN rutin, melyet akár parancs, akár program üzemmódban használhatunk. Formátuma megegyezik a LOAD utasításéval. A rutin el is indítja a betöltött programot. Ha erre nincs szükség, akkor a 36. sorba egy RTS utasítást kell írni.

A hanghatás rutinnal a felhasználót hanggal is figyelmeztetni lehet valamely tevékenységre. A hang hosszúságát a 49 279-es címen állíthatjuk be.

Hosszabb számítások, lemezműveletek esetére hasznos a keretszínező rutin, amely a megszakítást használja ki a színváltoztatásokra.

A kurzor pozicionálására már több program is megjelent. Ez annyival tud többet, hogy vizsgálja a határokat is, és hogy a kiírandó szöveget nem kell megadni. A szöveg egyébként a PRINT utasításnak felel meg.

A DATA BECKER könyvben megjelent rutin átírt változatával max. 255 karakter hosszúságú füzér olvasható be, miután megnyitottuk a megfelelő csatornát. Mivel egy GET sorozattal történik a beolvasás, előtte a megfelelő helyre pozicionálni kell.

RUBORIK BÉLA

```
LOAD+RUN......PAGE 0001

LINE# LOC   CODE        LINE

00001  0000                ;     LOAD+RUN
00002  0000                ;
00003  0000                ;HIVVASA SYS704"NEV"[,EGYSES]
00004  0000                ;CSAK BASIC PROGRAMRA ALKALMAZZA.
00005  0000                ;MASODLAGOS PARAMETERT NE ADJON!
00006  0000                ;
00007  0000                LKAPCS =$A
00008  0000                RUNMOD =$90
00009  0000                HEXPRG =$A533
00010  0000                INTERP =$A7AE
00011  0000                ERROR  =$E0F9
00012  0000                CLR    =$A659
00013  0000                LOAD   =$FFD5
00014  0000                STATUS =$FFB7
00015  0000                PARAM  =$E104
00016  0000                ERRPRI =$A437
00017  0000                *=704
00018  02C0  A9 00             LDA #0
00019  02C2  85 90             STA RUNMOD      ;PARANCS UZEMMOD
00020  02C4  85 0A             STA LKAPCS      ;LOAD KAPCSOLO
00021  02C6  20 D4 E1          JSR PARAM       ;LOAD PARAMETEREK
00022  02C9  A5 0A             LDA LKAPCS
00023  02CB  A6 2B             LDX $2B         ;BASIC KEZDOCIM
00024  02CD  A4 2C             LDY $2C         ;BEALLITASA
00025  02CF  20 D5 FF          JSR LOAD        ;LOAD RUTIN
00026  02D2  B0 19             BCS HIBA        ;HIBAS TOLTES?
00027  02D4  20 B7 FF          JSR STATUS      ;STATUS OLVASASA
00028  02D7  29 BF             AND #$BF        ;EOF BIT TORLESE
00029  02D9  F0 05             BEQ OKE         ;A VEGE RENDBEN?
00030  02DB  A2 1D             LDX #$1D        ;"LOAD ERROR"
00031  02DD  4C 37 A4          JMP ERRPRI      ;KIIRATASA
00032  02E0  86 2D       OKE   STX $2D         ;PROGRAM VEGENEK
00033  02E2  84 2E             STY $2E         ;BEALLITASA
00034  02E4  20 59 A6          JSR CLR         ;CHRSET ES CLR
00035  02E7  20 33 A5          JSR HEXPRG      ;PROGRAM OSSZELANCOLAS
00036  02EA  4C AE A7          JMP INTERP      ;INTERPRETER
00037  02ED  4C F9 E0    HIBA  JMP ERROR       ;I/O HIBA
00038  02F0                    .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 CLR     A659    ERROR   E0F9    ERRPRI  A437    HIBA    02ED
 INTERP  A7AE    LKAPCS  000A    LOAD    FFD5    HEXPRG  A533
 OKE     02E0    PARAM   E104    RUNMOD  0090    STATUS  FFB7

END OF ASSEMBLY
```

```
SOUND.....PAGE 0001

LINE# LOC   CODE        LINE

00001  0000                ;     HANGHATAS
00002  0000                ;
00003  0000                ;HIVVASA:SYS49200
00004  0000                ;
00005  0000                AF     = $0400
00006  0000                HE     = AF+24
00007  0000                HA     = AF+4
00008  0000                FL     = AF+5
00009  0000                MF     = AF+1
00010  0000                HF     = AF+6
00011  0000                *=49200
00012  C06C  78                   SEI
00013  C06D  A9 00                LDA #0
00014  C06F  A2 00                LDX #0
00015  C071  9D 00 D4      ZERO   STA AF,X
00016  C074  E8                   INX
00017  C075  E0 19                CPX #$19
00018  C077  D0 F8                BNE ZERO
00019  C079  A9 0F                LDA #$F
00020  C07B  8D 18 D4             STA HE
00021  C07E  A0 19                LDY #$19
00022  C080  A2 FF       FOR    LDX #$FF
00023  C082  A9 40       FOR1   LDA #$40
00024  C084  8D 06 D4             STA HF
00025  C087  A9 21                LDA #$21
00026  C089  8D 04 D4             STA HA
00027  C08C  A9 80                LDA #$80
00028  C08E  8D 05 D4             STA FL
00029  C091  A9 24                LDA #$24
00030  C093  8D 01 D4             STA MF
00031  C096  A9 A9                LDA #$A9
00032  C098  8D 00 D4             STA AF
00033  C09B  CA                   DEX
00034  C09C  E0 00                CPX #0
00035  C09E  D0 E2                BNE FOR1
00036  C0A0  88                   DEY
00037  C0A1  C0 00                CPY #0
00038  C0A3  D0 DE                BNE FOR
00039  C0A5  8E 04 D4             STX HA
00040  C0A8  8E 05 D4             STX FL
00041  C0AB  58                   CLI
00042  C0AC  60                   RTS
00043  C0AD                       .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 AF      0400    FL      0405    FOR     C080    FOR1    C082
 HA      0404    HE      0418    HF      0406    MF      0401
 ZERO    C071

END OF ASSEMBLY
```

```
KERETSZINEK.....PAGE 0001

LINE# LOC   CODE        LINE

00001  0000                ;     GYORS KERETSZIN VALTHSOK
00002  0000                ;DEKAPCSOLAS SYS820
00003  0000                ;KIKAPCSOLAS SYS820+29
00004  0000                ;
00005  0000                IROVEC =$314
00006  0000                IRQ    =$EA31
00007  0000                BORDER =$D020
00008  0000                *=$334
00009  0334  A9 45               LDA #<UJCIM     ;SAJAT RUTINUNK
00010  0336  A9 03               LDY #>UJCIM     ;CIMENEK BETOLTESE
00011  0338  8D 14 03            STA IROVEC      ;A MEGSZAKITAS
00012  033B  8C 15 03            STY IROVEC+1    ;VEKTORNEK
00013  033E  A9 D0               LDA BORDER      ;EREDETI KERETSZIN
00014  0341  8D 63 03            STA TAROLO      ;ELMENTESE
00015  0344  60                  RTS
00016  0345  AD 62 03     UJCIM  LDA RAMKA       ;A KERET SZINENEK
00017  0348  8D 20 D0            STA BORDER      ;EGYIVEL VALO
00018  034B  EE 62 03            INC RAMKA       ;NOVELESE
00019  034E  4C 31 EA            JMP IRQ         ;VISSZA A MEGSZAKITASBA
00020  0351  A0 31        KIKAPC LDA #<IRQ       ;A MEGSZAKITAS
00021  0353  A0 EA               LDY #>IRQ       ;VEKTORAINAK
00022  0355  8D 14 03            STA IROVEC      ;VISSZAALLITASA
00023  0358  8C 15 03            STY IROVEC+1    ;
00024  035B  AD 63 03            LDA TAROLO      ;KERETSZIN
00025  035E  8D 20 D0            STA BORDER      ;VISSZAALLITASA
00026  0361  60                  RTS             ;
00027  0362  00           RAMKA  .BYTE 0
00028  0363  00           TAROLO .BYTE 0
00029  0364                       .END

ERRORS = 00000
```

```
SYMBOL TABLE

SYMBOL VALUE
 BORDER   D020    IRQ      EA31    IRQVEC   0314    KIKAPC   0351
 MUNKA    0362    TAROLO   0363    UJCIM    0345

END OF ASSEMBLY



CURSOR......PASE 0001

LINE# LOC   CODE        LINE


00001  0000                ;     CURSOR POZICIONALAS
00002  0000                ;
00003  0000                ;HIVASA:SYS870,X,Y[,....]
00004  0000                ;
00005  0000                CHKCOM =$AEFD
00006  0000               -GETBYT =$B79E
00007  0000                CURSOR =$FFF0
00008  0000                ILLEG  =$B248
00009  0000                CHRGOT =$79
00010  0000                PRINT  =$AAA4
00011  0000                *=$70
00012  0366  20 FD AE            JSR CHKCOM
00013  0369  20 9E B7            JSR GETBYT      ;X PARAMETER
00014  036C  8A                  TXA
00015  036D  C9 28               CMP #$28
00016  036F  10 1D               BPL ERROR       ;HA X>39
00017  0371  48                  PHA
00018  0372  20 FD AE            JSR CHKCOM
00019  0375  20 9E B7            JSR GETBYT      ;Y PARAMETER
00020  0378  E0 19               CPX #$19
00021  037A  10 12               BPL ERROR       ;HA Y>24
00022  037C  68                  PLA
00023  037D  A8                  TAY
00024  037E  18                  CLC             ;POZICIONALAS
00025  037F  20 F0 FF            JSR CURSOR
00026  0382  20 79 00            JSR CHRGOT
00027  0385  F0 06               BEQ KILEP       ;HA NINCS TOBB
00028  0387  20 FD AE            JSR CHKCOM
00029  038A  4C A4 AA            JMP PRINT       ;PRINT-RE UGRAS
00030  038D  60          KILEP   RTS
00031  038E  4C 48 B2    ERROR   JMP ILLEG       ;"ILLEGEL QUANTITY"
00032  0391                      .END


ERRORS = 00000


SYMBOL TABLE

SYMBOL VALUE
 CHKCOM   AEFD    CHRGOT   0079    CURSOR   FFF0    ERROR    038E
 GETBYT   B79E    ILLEG    B248    KILEP    038D    PRINT    AAA4


END OF ASSEMBLY


INPUT......PASE 0001

LINE# LOC   CODE        LINE


00001  0000                ;     INPUT
00002  0000                ;
00003  0000                ;HIVASA
00004  0000                ;SYS920,CSATORNA,HOSSZ,VALTOZO
00005  0000                ;
00006  0000                CHKCOM =$AEFD
00007  0000                FHDVAR =$B08B
00008  0000                FRESTR =$B6A3
00009  0000                VARADR =$49
00010  0000                STRRES =$B475
00011  0000                PARA   =$61
00012  0000                BASIN  =$E112
00013  0000                CLRCH  =$FFCC
00014  0000                GETBYT =$B79E
00015  0000                CHRIN  =$E11E
00016  0000                *=$39
00017  033C  20 FD AE            JSR CHKCOM      ;CSATORNA
00018  033F  20 9E B7            JSR GETBYT
00019  0342  20 1E E1            JSR CHRIN
00020  0345  20 FD AE            JSR CHKCOM
00021  0348  20 9E B7            JSR GETBYT      ;HOSSZ
00022  034B  8A                  TXA             ;ELMENTESE
00023  034C  48                  PHA
00024  034D  20 FD AE            JSR CHKCOM
00025  0350  20 8B B0            JSR FHDVAR      ;VALTOZO KERESESE
00026  0353  85 49               STA VARADR
00027  0355  84 4A               STY VARADR+1
00028  0357  20 A3 B6            JSR FRESTR
00029  035A  68                  PLA
00030  035B  20 75 B4            JSR STRRES      ;TERULET FOGLALAS
00031  035E  A0 02               LDY #2
00032  0360  B9 61 00    STORE   LDA PARA,Y      ;KARAKTEREK BEOLVASASA
00033  0363  91 49               STA (VARADR),Y
00034  0365  88                  DEY
00035  0366  10 F8               BPL STORE
00036  0368  C8                  INY
00037  0369  20 12 E1    FETCH   JSR BASIN
00038  036C  91 62               STA (PARA+1),Y
00039  036E  C8                  INY
00040  036F  C4 61               CPY PARA
00041  0371  D0 F6               BNE FETCH
00042  0373  20 CC FF            JSR CLRCH
00043  0376  60                  RTS
00044  0377                      .END


ERRORS = 00000
```

```
SYMBOL TABLE

SYMBOL VALUE
 BASIN    E112    CHKCOM   AEFD    CHRIN    E11E    CLRCH    FFCC
 FETCH    0369    FHDVAR   B08B    FRESTR   B6A3    GETBYT   B79E


SYMBOL TABLE

SYMBOL VALUE
 PARA     0061    STORE    0360    STRRES   B475    VARADR   0049


END OF ASSEMBLY
```

```
10 rem       loadrun
100 fori=704to749:readx:pokei,x:s=s+x:next
110 ifs<>5182thenprint"hiba a data sorokban!"
1000 data 169,0,133,157,133,10,32,212
1010 data 225,165,10,166,43,164,44,32
1020 data 213,255,176,25,32,183,255,41
1030 data 191,240,51,162,29,76,55,164
1040 data 134,45,132,46,32,89,166,32
1050 data 51,165,76,174,167,76

ready.
```

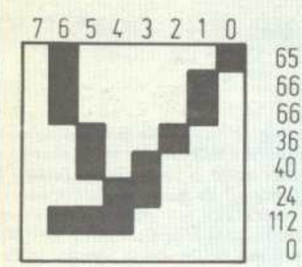
```
10 rem      sound
100 fori=4928to49324:readx:pokei,x:s=s+x:next
110 ifs<>84932thenprint"Hiba a data sorokban!"
1000 data 120,169,0,162,0,157,0,212
1010 data 232,224,25,208,248,169,15,141
1020 data 24,212,169,25,162,255,169,64
1030 data 141,6,212,169,33,141,4,212
1040 data 169,128,141,5,212,169,36,141
1050 data 1,212,169,169,141,0,212,202
1060 data 224,0,208,226,136,192,0,208
1070 data 219,142,4,212,142,5,212,88
1080 data 96

ready.
```

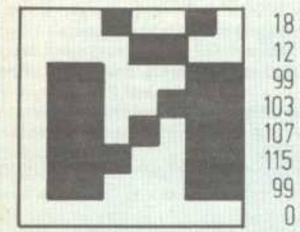
```
10 rem   keratszinezes
100 fori=820to867:readx:pokei,x:s=s+x:next
110 ifs<>4526thenprint"hiba a data sorokban!"
1000 data 169,69,160,3,141,20,3,140
1010 data 21,3,173,32,208,141,99,3
1020 data 96,173,99,3,141,32,208,238
1030 data 99,3,76,49,234,169,49,160
1040 data 234,141,20,3,140,21,3,173
1050 data 99,3,141,32,208,96,0,0

ready.
```

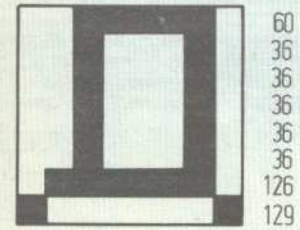
```
10 rem    cursor pozicionalas
100 fori=870to910:readx:pokei,x:s=s+x:next
110 ifs<>4706thenprint"hiba a data sorokban!"
1000 data 32,253,174,32,158,183,138,201
1010 data 40,16,29,72,32,253,174,32
1020 data 158,183,224,25,16,10,104,168
1030 data 24,32,240,255,32,121,0,240
1040 data 6,32,253,174,76,164,170,96
1050 data 76

ready.
```

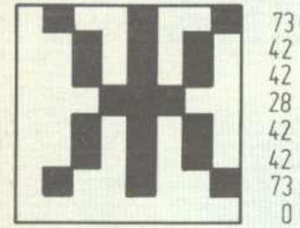
```
10 rem    input
100 fori=920to986:readx:pokei,x:s=s+x:next
110 ifs<>7580thenprint"hiba a data sorokban!"
1000 data 32,253,174,32,158,183,32,89
1010 data 225,32,253,174,32,158,183,138
1020 data 72,32,253,174,32,139,176,133
1030 data 73,132,74,32,163,162,104,32
1040 data 117,180,160,2,185,97,0,145
1050 data 73,136,16,248,200,32,10,225
1060 data 145,90,200,196,97,208,246,32
1070 data 204,255,96

ready.
```

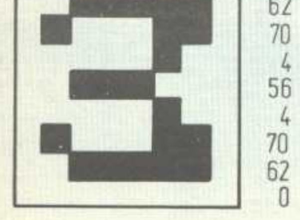
Cirill betűk











Programozható karakterek előállítási módjáról több szakkönyvben és e lap korábbi számaiban is olvashatunk. De az e témában kevésbé járatosaknak nem kis feladatot jelent egy ilyen program megírása.

Az orosz nyelvű szövegek számítógépes felhasználásához elengedhetetlen a cirillbetűs írás. A listán látható program a nagybetűs latin karakterek mellett a cirill nagybetűs írásra is lehetőséget nyújt.

A program felépítése

Minden egyes karaktert egy 8×8-as pontrács határoz meg. Egy-egy karaktert 8 egymást követő ROM-beli címen definiálnak azáltal, hogy az egyes bitek ki- vagy bekapcsolt állapotban vannak-e. Új karakterhez csak valamely régi megváltoztatása útján juthatunk. A ROM-beli címek bitjeit megváltoztatni nem tudjuk, csak átmásolni a RAM területére, ahol a módosítás megvalósítható.

- **10-100** Képernyőtörlés, színbeállítás, feliratok.
- **110** A megszakítások letiltása.
- **120** A B/K regiszterek kikapcsolása, a ROM karakterek bekapcsolása.
- **130-160** A 8192—10239 címekre átmásoljuk a ROM 53248—55295 tartalmát.
- **170-180** A B/K visszakapcsolása és a megszakítások engedélyezése.
- **190** Ahhoz, hogy a továbbiakban a karakterjeleket a 8192-es címtől olvassa ki a számítógép, a karaktergenerátor kezdetét be kell állítani. Ezután már lehetőségünk van az új karakterek előállítására.
- **200-400** A DATA sorok kiolvasásával rendre előállnak a megfelelő új betűk. A RESTORE utáni rész az inverz előállításokat végzi.

A DATA sorok készítési módját az *ábráról* leolvashatjuk. Amely pontokat ki akarjuk gyújtani, azok oszlopsorszám szerinti kettőhatványait összeadjuk. Így az ábra első értékét 64+1 alapján kaptuk.

Ha most egy NEW parancsot kiadunk, a programunk ugyan elvész, de az új karaktereket a továbbiakban is használni tudjuk. Futtathatunk új programot is, de hogy a nyert karaktereinket el ne rontsuk, a BASIC-terület kezdetét át kell állítani a 490-500-as sorok utasítása szerint.

Az új karaktereket a RUN—STOP/RESTORE billentyűzettel törölni tudjuk. Ha a számítógépet kiindulási helyzetbe akarjuk hozni kikapcsolás nélkül, akkor parancsban írjuk be: POKE 43,1 : POKE 44,8 : POKE 53272,21.

Minden cirillbetűt a SHIFT és a latin megfelelőjének billentyűzetével írhatunk a képernyőre. Amelyiknek nincs latin megfelelője, azt ugyancsak a SHIFT és az alábbi táblázat alapján kapjuk:

+	Щ
—	Ь
*	Ч
:	Ъ
;	Ы
,	Ю
.	Я
Q	Ё
W	Е
E	Э
Y	С
X	Ж

SIMON GYULA

```
10 [illegible]
20 REM ********************
30 REM *                  *
40 REM *    CIRILL BETUK  *
50 REM *                  *
60 REM ********************
70 PRINT  CHR$(147)CHR$(17)CHR$(17)
80 PRINTTAB(14)"CIRIL BETUK"
90 PRINT:PRINT:PRINT
100 PRINT " VARJON EGY PERCET"
110 POKE 56334,PEEK(56334) AND 254
120 POKE 1,PEEK(1) AND 251
130 FOR I= 0 TO 255
140 FOR J=0 TO 7
150 POKE8192 +I*8+J,PEEK(53248+I*8+J)
160 NEXTJ:NEXTI
170 POKE1,PEEK(1) OR 4
180 POKE56334,PEEK(56334) OR 1
190 POKE53272,(PEEK(53272) AND 240)+8
200 FOR C=60 TO 93
210 FOR BY= 0 TO 7
220 READ N
230 POKE 8192 +(8*C)+BY,N
240 NEXTBY  :NEXTC
250 FOR A=27 TO 29 STEP 2
260 FOR B=0 TO 7
270 READ N
280 POKE 8192+8*A +B,N
290 NEXT B : NEXT A
300 RESTORE
310 FOR C=60+128TO 93+128
320 FOR BY= 0 TO 7
330 READ N
340 POKE8192 +(8*C)+BY,255-N
350 NEXTBY  :NEXTC
360 FOR A=27 TO 29 STEP 2
370 FOR B=0 TO7
380 READ N
390 POKE  8192+8*(128+A)+B,255-N
400 NEXTB :NEXTA
410 PRINT:PRINT
420 PRINT"A KARAKTEREK MODOSITASA KESZ"
430 PRINT
440 PRINT
450 PRINT"A BASIC KEZDO CIMENEK ";
460 PRINT"ATALLITASA CELJA- BOL IRJA";
470 PRINT" BE PARANCSBA "
480 PRINT:PRINT
490 PRINTCHR$(18)" POKE43,1: POKE44,40";
500 PRINTCHR$(18)"  POKE10240,0:NEW  "
510 END
520 DATA 68,78,91,113,81,74,68,0
530 REM < JU
540 DATA  0,127,127,0,127,127,0,0:REM =
550 DATA 126,70,70,126,38,70,134,0
560 REM > JA
570 DATA 60,34,6,12,24,0,24,0: REM ?
580 DATA 102,102,102,102,62,6,6,0
590 DATA 8,28,20,54,34,127,65,0:REM A
600 DATA 127,97,96,126,99,99,126,0:REM B
610 DATA 70,70,70,70,70,70,126,1:REM C
620 DATA 60,36,36,36,36,36,126,129:REM D
630 DATA 60,6,6,30,6,6,60,0:REM E
640 DATA 8,62,107,73,107,62,8 ,0 REM F
650 DATA 127,97,96,96,96,96,96,0:REM G
660 DATA 99,54,28,8,8,54,99,0:REM H
670 DATA 0,71,75,75,83,83,99,0:REM I
680 DATA 18,12,99,103,107,115,99,0
690 REM J
700 DATA 99, 100,104,112,104,100,99,0
710 REM K
720 DATA 31,19,19,19,35,35,99,0:REM L
730 DATA  99,119,119,107, 99, 99, 99,0
740 REM M
750 DATA 99,99,99,127,99,99,99,0:REM N
760 DATA 62, 99, 99, 99, 99, 99,62,0
770 REM O
780 DATA 127,99,99,99,99,99,99,0:REM P
790 DATA 10,10,62,48,60,48,62,0:REM JO Q
800 DATA 126,97,97,124,96,96,96,0:REM R
810 DATA  73, 73, 73, 73, 73, 73,127,0
820 REM S
830 DATA 126,126,24,24,24,24,24,0:REM T
840 DATA  67, 67, 67,102,108,28,240,0
850 REM U
860 DATA 126,97,97,124,97,97,126,0:REM V
870 DATA 127,97,96,126,96,97,127,0
875 REM JE W
880 DATA 73,42,42,28,42,42,73,0:REM ZS X
890 DATA 62,99,96,96,96,99,62,0:REM SZ Y
900 DATA 62,70,4,56,4,70,62,0:REM Z
910 DATA 146,146,146,146,146,146,254,1
920 REM SCSA +
930 DATA 65,65,65,121,69,69,121,0:REM UE
940 DATA 64,64,64,120,68,68,120,0
950 REM LAGY JEL -
960 DATA 224,160,32,47,49,49,54,0
970 REM KEMENY JEL
980 DATA 65,65,65,121,69,69,121,0
990 REM UE

READY.
```

Néhány képernyő-kezelő rutin

A program először helyet biztosít a gépi kód számára, majd az öt rutint elhelyezi a 6639-es címtől kezdődően. A rutinok feladata sorrendben a következő:

1. EXT 0 utasítás: a teljes képernyő tartalmát jobbra lépteti.
2. EXT 1 utasítás: a teljes képernyő tartalmát balra lépteti.
3. EXT 2 utasítás: a teljes képernyő tartalmát felfelé lépteti.
4. EXT 3 utasítás: a teljes képernyő tartalmát lefelé lépteti.
5. EXT 4 utasítás: a papír és a tinta színét állítja be. A rutinok az ötödik kivételével a botkormány mozgatásával aktivizálódnak.

A függőleges mozgatás egyszerű blokkáthelyezés. Ebből következik, hogy a mozgatás gyorsítható.

Felfelé léptetés gyorsítása: a H regiszterpár tartalmának növelése 64-gyel vagy többszörösével, a BC regiszterpár tartalmának csökkentése ugyanannyival.

Lefelé léptetés gyorsítása: a HL és BC regiszterpárok tartalmának csökkentése 64-gyel vagy többszörösével.

Az ötödik rutinban a 100-as, illetve 101-es portcímekre más adatok írásával a papír, illetve a tinta színe megváltozik.

A gép mellé kapott segédletekből kiderül, hogy a memória 80 kbájt méretű lehet. Emiatt a VIDEO RAM-ot gépi kódban be kell „lapozni". Erre szolgál a DATA sorok elején szereplő 62,80,211,2 kódsorozat. Magyarul: a 2-es portcímre 80-at kell beírni.

A rutinok könnyen átírhatók úgy, hogy a képernyőtartalomnak csak egyes részei mozduljanak el.

GULYÁS KIS PÉTER

```
10 LOMEM 6800
20 GRAPHICS 2
30 GOSUB 500
35 REM USRTAB táblázat feltöltése
40 POKE 33,239:POKE 34,25
50 POKE 35,8:POKE 36,26
60 POKE 37,33:POKE 38,26
70 POKE 39,51:POKE 40,26
80 POKE 41,69:POKE 42,26
85 REM Gépi kód betöltése
90 FOR A=0 TO 96
100 READ B
110 POKE 6639+A,B
120 NEXT A
130 DATA
    243,62,80,211,2,33,255,127,6,240,14,64,35,
    203,30,13,32,250,55,63,5,32,243,251,201
140 DATA
    243,62,80,211,2,33,0,192,6,255,14,64,43,203,
    22,13,32,250,55,63,5,32,243,251,201
150 DATA
    243,62,80,211,2,33,64,128,17,0,128,1,192,63,
    237,176,251,201
160 DATA
    243,62,80,211,2,33,191,191,17,255,191,1,
    192,63,237,184,251,201
170 DATA 243,62,5,211,100,62,8,211,101,251,201
    180 SET DELAY 1:SET RATE 1
190 EXT 4
195 REM Billentyűzetfigyelés
200 A$=INKEY$
210 IF A$=CHR$(4) THEN EXT 0
220 IF A$=CHR$(19) THEN EXT 1
230 IF A$=CHR$(5) THEN EXT 2
240 IF A$=CHR$(24) THEN EXT 3
250 IF A$="v" THEN SET DELAY 30:SET RATE
    3: END
260 GOTO 200
500 PLOT
    300,300:300,700:700,700,700,300:300,300:PL-
    OT,
    304,304,PAINT
510 RETURN
```

ZX–Spectrum

A Beta basic és alkalmazása I.

A cikksorozat azt mutatja be, hogy a Beta basic 1.8 felhasználói program segítségével hogyan írhat magának bárki olyan hasznos programot, amely segítségére lehet a nyelvtanulásban vagy feljegyzések tárolásában, és amelyet a Beta basic nélkül csak gépi kódban tudna megírni. A program egy szótár. A beírt — tetszőleges hosszúságú, de max. 255 karakteres — szavak hosszától függően kb. 1000-1200 szót képes gyorsan visszakereshető módon tárolni.

A megtanult szavak egyenként kitörölhetők, helyet szabadítva fel az új ismeretlen szavak számára. A mindenkori szóállomány a programból magnóra menthető és onnan bármikor visszatölthető.

A Beta basic

A Beta basic az angliai Betasoft Ltd. fejlesztésének eredménye. Több változata terjedt el. Én most csak az 1.8 verzióval foglalkozom, mivel programot ezen írtam.

A Beta basic 1.8 34 új utasításszót és 19 új függvényt tesz hozzá a gép BASIC-jéhez. A kulcsszavak grafikus üzemmódban 1 billentyű lenyomására, a függvények az FN // segítségével érhetők el. Shift nélküli kulcsszó az EDIT, ami akkor íródik ki, ha Enter után nyomjuk meg a 0-t. (A program írói úgy okoskodtak, hogy 0-val kezdődő sor nem létezik.)

A Beta betöltése: LOAD "" ENTER. Ekkor betöltődik a háromsoros BASIC-betöltő és a 9,5 kbájt terjedelmű gépi kódú rész. Ha a betöltés sikeres volt, megjelenik a "Beta basic 1.8 c Betasoft 1984" felirat.

A BASIC-betöltő 0-dik sora tartalmazza az új függvényeket; ez a programban marad, és LIST 0 parancsra láthatóvá válik. Az első sor a Beta esetleges másolását segíti, a második tölti be a gépi kódot, meghívja, és kitörli az első sort és saját magát.

A Betában írt BASIC-programok magnóra mentésével viszik magukkal a 0-dik sort is, így visszatöltésnél használható a LOAD. Ha egy már meglévő Spectrum BASIC-ben írt programot akarunk átalakítani Betára, akkor ezt MERGE utasítással töltsük be, különben elveszítjük a függvényeket.

A Beta basicban írt programok betöltése előtt be kell küldenünk a Betát is, különben listázásnál az új kulcsszavak helyén csak magányos grafikus karaktereket találunk, és futtatásnál a program "C Nonsense in BASIC" üzenettel leáll.

Az 1.8-as verzió csak 48 k-s gépen futtatható. A gépi kódú rész az 55801-es címnél kezdődik és 65367-ig tart. Tehát az UDG terület használható. Ilyenkor azonban ki kell kapcsolnunk a Betát KEYWORDS 0 utasítással ("G" kurzor, 8). Ekkor minden grafikus karaktert használhatunk, kivéve a 8. billentyűn lévőt, amelyre a visszatéréshez van szükség, hogy kiadhassuk a KEYWORDS 1 utasítást.

Mivel a Beta basic leírása kb. 35 gépelt oldal, csak a szótárprogramban lévő utasításokat és függvényeket magyarázom el. Előbb azonban még egy ötlet. Ha már van egy Beta basicben írt programunk, amit sűrűn szeretnénk használni, de fárasztónak tartjuk, hogy a Betát előtte mindig külön szalagról be kell tölteni, megtehetjük a következőket. Betöltjük a gépbe a Betát és a programot. Programunkat kiegészíthetjük az alábbi sorokkal:

1 CLEAR 55800 : LOAD "beta" CODE : RANDOMIZE USR 58419.
2 CLS : DELETE 1 TO 2

Ezután fogunk egy kazettát, és kimentjük rá programunkat:

SAVE "program" LINE 1

Most kiadjuk a RANDOMIZE USR 59904 parancsot, és közvetlenül a program után kimentjük kazettánkra a Beta gépi kódját is:

SAVE "beta" CODE 55801,9567

Így ha legközelebb használni akarjuk programunkat, nincs más dolgunk, csak a kazettát betenni a magnóba és kiadni a LOAD "" parancsot. Ekkor betöltődik a program, 0-dik sora viszi magával az új függvényeket, és autostarttal elindul. Első sora beállítja a RAMTOP-ot 55800-ra, betölti a Beta basicet és aktivizálja (RAND USR 58419).

A második sor letörli a copyright feliratot, majd kitörli a programból az első sort és saját magát (DELETE 1 TO 2). Ezután már zavartalanul futhat a program, de csak abban az esetben, ha a futás közben semmilyen hibaüzenet vagy hibakezelés nem fordul elő. Mert ha igen, akkor a Beta végre szóhoz juthat, és boldogan követelheti a copyright üzenet után neki kijáró ENTER-t. Ilyenkor a program nem hibaüzenettel, hanem villogó "K" kurzorral áll le, és egy RUN parancs után már normálisan fut tovább. Ezért ha programunkban számítunk valamilyen hibaüzenetre, jobban teszszük, ha a lista elején később kitörlődő sorok valamelyikébe beírunk egy STOP utasítást.

A RAND USR 59904 lényegében teljesen kikapcsolja a Beta basicet. Csak annyi köze lesz a géphez, hogy benne van a memóriában. A Beta akkor is kimenthető, ha nem adjuk ki ezt a parancsot, de VERIFY esetén még hibátlan felvételnél is "R Tape loading error" üzenetet kapunk. Gondolom, a Betasoft emberei ezzel is az illegális programmásolók életét akarták keseríteni.

Utasítások

DELETE ("G" 7)

Alakja DELETE sorszám TO sorszám.

Az adott határok közötti programszakasz valamennyi sorát törli. Ha magát az utasítást is törölni akarjuk, akkor a DELETE legyen a törölni kívánt blokk utolsó utasítása. Például:
DELETE TO 60 ; A 0-dik sort kivéve, 60-ig mindent töröl
DELETE 60 TO 80; 60 és 80 között mindent töröl
DELETE 80 TO ; 80 felett mindent töröl

DO ("G" D), **EXIT IF** ("G" I), **LOOP** ("G" L)

Egy a FOR—NEXT-nél rugalmasabban használható ciklusszerkezet. A ciklus kezdőpontja a DO, végpontja a LOOP. Tesztelhető az elején (DO WHILE, DO UNTIL), a végén (LOOP WHILE, LOOP UNTIL), vagy elhagyható a közepéről (EXIT IF, ha egy feltétel igazzá válik). Ha a ciklusszerkezetből a LOOP hiányzik, "S Missing LOOP", ha a DO, "T LOOP without DO" hibaüzenetet kapunk.

GET ("G" G)

Az INKEY$-hez hasonlóan ENTER lenyomása nélkül vihetünk be adatot a billentyűről. A különbség az, hogy a program mindaddig várakozik, míg valamilyen billentyűt le nem nyomtunk. Sztringváltozót használva egy egykarakteres sztringet, numerikus változónál pedig a billentyűre írt számnak megfelelő értéket kapunk. Betűk lenyomásakor A=10, B=11 stb.

ON ("G" O)

A listaszerűen felsorolt sorszámok valamelyikére enged meg GOTO vagy GOSUB jellegű ugrást, aszerint, hogy az utána álló numerikus változó értéke mekkora. Például INPUT opció: GOTO ON opció; 100, 30, 52, 200 stb.

PLOT (a szokott helyen) (Plot x,y; a$ · X,y= az első betű bal felső sarka)

Beta basicben lehetőségünk van arra, hogy PLOT utasítással egy tetszőleges szöveget (táblázat, diagram, alsó, felső index stb.) kiírhassunk a képernyőre.
Próbáljuk ki a következőt:
LET a$="szöveg":FOR n=210 TO 0 STEP- 1: PLOT n,88;a$: NEXT n

POKE (a szokott helyen) (POKE cím, a$)

Egyetlen POKE utasítással egy egész sztringet vihetünk a memóriába, az adott címtől kezdődően. Az INSTRING és MEMORY$ függvényekkel kombinálva, ez lehetővé teszi az egész memória villámgyors átmozgatását, vizsgálatát. Így lehet például képállományokat létrehozni és azokat váltogatni a képernyőn (LDIR kiváltás).

POP ("G" Q)

A programban egy 2 bájt nagyságú címet távolít el a GOSUB verem tetejéről, megakadályozva annak túlcsordulását.

DEF PROC ("G" 1), **PROC** ("G" 2), **END PROC** ("G" 3)

Procedure (eljárás): vagyis lehetőség nyílik különböző eljárások definiálására és ezek meghívására a program tetszőleges helyéről. Az eljárás vehető egy névvel ellátott szubrutinnak is. Előnye, hogy nem kell törődni azzal, hogy a program rá ne fusson. Az eljárás elejét a DEF PROC jelzi. Ez az adott sor első utasítása kell hogy legyen. Az elnevezés bármilyen lehet, használhatunk egy már meglévő változónevet is. Egyetlen kikötés, hogy az első karakter betű legyen. Az eljárás tetszőleges hosszúságú lehet, a végét az END PROC jelzi. Meghívása: PROC.

Az eljárás közepéből a szubrutinhoz hasonlóan nem illik kiugrani. Ha mégis megtesszük, adjunk utána POP utasítást, hogy a visszatérési címet kitöröljük a GOSUB veremből. Hibák: PROC hívás definiálás nélkül vagy END PROC DEF PROC nélkül: W Missing DEF PROC.

```
  10 CLEAR
  15 ON ERROR 15
  20 PRINT AT 1,10;"S Z O T A R"
  25 PRINT AT 2,2;"_____________
_____"
  30 PRINT AT 5,7;"1.   Beiras";
AT 7,7;"2.   Kereses";AT 9,7;"3.
   Kimentes";AT 11,7;"4.   Betol
tes";AT 13,7;"5.   Szo torlese";
AT 15,7;"6.   File torles";AT 17
,7;"7.   Informacio"
  35 PRINT AT 20,2;"_____________
_____"
  40 PRINT )0,AT 0,11;"  1986":
 GET a
  45 CLS : GO TO  ON a;100,200,3
00,400,800,55,700
  50 GO TO 10
  55 PRINT )0,AT 1,1;"Torleshez
 nyomd meg az ENTER-t!": PAUSE 0
  60 LET r$=INKEY$: IF r$<>CHR$
13 THEN  GO TO 70
  65 CLEAR : GO SUB 500: PRINT A
T 19,8;"FILE TOROLVE!": GO TO 15
  70 GO TO 10
 100 INPUT AT 0,0;" BEIRAS:";TAB
 16;" Visszateres:0";AT 2,0; LIN
E b$
 105 IF b$="0" THEN  GO TO 10
 110 PROC beiras
 115 PRINT b$
 120 GO TO 100
 200 INPUT AT 1,0;" KERESES:";AT
 1,16;" Visszateres:0";AT 3,0; L
INE k$
 205 IF k$="0" THEN  GO TO 10
 210 PRINT  PAPER 0; INK 7;k$
 215 ON ERROR 200: PROC kereses
 220 POP : GO TO 200
 300 INPUT AT 0,0;" KIMENTES:";T
AB 15;"Filenev:";AT 2,0;m$
 305 GO SUB 600
 310 SAVE m$CODE 28700,cim-28700
 315 ON ERROR 300
 320 INPUT " VERIFY? (i/n)";v$
 325 IF v$="n" THEN  GO TO 10
 330 VERIFY m$CODE
 335 GO TO 10
 400 GO SUB 600: IF cim<>28700 T
HEN  PRINT AT 10,1;"Betoltes  el
 ott  a  szoallomanytorlesehez  h
 asznald  a  menu 6.";TAB 12;"opc
 ojat.": PRINT )0,AT 1,3;"Nyomj m
 eg egy billentyut!": PAUSE 0: GO
 TO 10
 405 INPUT AT 0,0;" BETOLTES: ";
TAB 15;"Filenev:";AT 2,0;m$
 410 ON ERROR 400
 415 LOAD m$CODE
 420 GO TO 10
 500 RANDOMIZE USR 55783
 505 RETURN
 600 LET cim=INSTRING(28700,MEMO
RY$(),CHR$ 0)
 605 RETURN
 700 GO SUB 600: CLS : PRINT AT
1,6;"I N F O R M A C I O";AT 5,1
 ;"A tar merete:";TAB 20;"27000
byte";AT 7,1;"Felhasznalva: ";TA
B 20;cim-28700;" byte":AT 9,1;"S
zabad:";TAB 20;55760-cim;" byte"
 705 PLOT 0,50: DRAW 255,0: DRAW
 0,5: DRAW -255,0: DRAW 0,-5
 710 PLOT 0,50: DRAW 0,-4: PLOT
128,50: DRAW 0,-3: PLOT 255,50:
DRAW 0,-4
 715 PLOT 0,52: DRAW (cim-28700)
/100,0: DRAW 0,1: DRAW -(cim-287
00)/100,0
 720 PLOT 0,40;"28700": PLOT 215
,40;"55760"
 725 LET a=USR 55760: PRINT AT 1
1,1;"Beirva:";TAB 20;a;" szopar"
 730 ON ERROR 200
 735 PRINT )0,AT 1,3;"Nyomj meg
egy billentyut!": PAUSE 0: GO TO
 10
 800 INPUT AT 1,0;" SZOTORLES:";
TAB 16;"Visszateres:0";AT 3,0; L
INE k$
 805 IF k$="0" THEN  GO TO 10
 810 PRINT  INVERSE 1;k$: ON ERR
OR 800: PROC szotorles
 815 POP : GO TO 800
1000 DEF PROC beiras
1005 GO SUB 600
1010 IF 55759-cim<=LEN b$+1 THEN
  PRINT "A szotar megtelt.": GO
TO 200
1015 POKE cim,42
1020 POKE cim+1,b$
1025 END PROC
1100 DEF PROC kereses
1105 LET c=INSTRING(28700,MEMORY
$()( TO 55760),k$)
1110 IF c=0 THEN  PRINT "Nem ism
erek ilyen szot.": LET h=0: GO T
O 1175
1115 LET h=1
1120 DO
1125 EXIT IF PEEK c=42
1130 LET c=c-1
1135 LOOP
1140 LET c1=c
1145 LET c2=INSTRING(c+1,MEMORY$
()( TO 55760),CHR$ 42)
1150 IF c2=0 THEN  GO SUB 600: L
ET c2=cim
1155 LET i$=MEMORY$()(c1+1 TO c2
-1)
1160 PRINT i$: PAUSE 20
1165 LET c=INSTRING(c2,MEMORY$()
( TO 55760),k$)
1170 IF c<>0 THEN  GO TO 1120
1175 END PROC
1200 DEF PROC szotorles
1205 PROC kereses: IF h=0 THEN
GO TO 1245
1210 PRINT )0,AT 1,3;"Az utolso
szo torlese: ""1""": GET a
1215 GO TO  ON a;1225
1220 GO TO 1245
1225 GO SUB 600: POKE c1,STRING$
(LEN i$+1,CHR$ 0)
1230 POKE c1,MEMORY$()(c2 TO cim
)
1235 GO SUB 600: POKE cim+1,STRI
NG$(LEN i$,CHR$ 0)
1240 PRINT "Az utolso szo torolv
e!"
1245 END PROC
```

ON ERROR ("G" N)

Mikor ez az utasítás üzembe lép, a program szubrutinként ugrik az ON ERROR után megadott címre, ha egyébként valamilyen hibaüzenetet kapnánk. Itt aztán tesztelhetjük a hibát; erre rendelkezésünkre áll három változó, amelyeket karakterenként kell begépelni. A line és a stat változók annak a sornak és soron belül az utasításnak a számát tartalmazzák, ahol a hiba bekövetkezett. Az error változó a hiba típusát jelzi. Az 1—8 hibáknál értéke 1—8, A-tól kezdve pedig 10, 11 stb.

A hibakezelő szubrutinból visszatérés RETURN vagy POP utasítással történik. Így minden hiba kezelhető, kivéve a 0, OK és a 9 STOP.

A leírás szerint kerülendő az olyan eljárás, amilyet én a programban alkalmaztam: vagyis a hiba nincs szigorúan definiálva, és a hibakezelés szubrutinra téve. A GOSUB verem lassan, de biztosan telítődne, ha nem törölné minden menühöz való visszatérésnél teljesen a 10 CLEAR sor, valamint címenként a 220 és a 815 POP sorok. Ezzel a módszerrel tudtam elérni, hogy keresésnél a scroll?—BREAK válaszra ne álljon le a program. Ez a megoldás azért veszélyes, mert nem veszünk észre egy esetleges hibát, vagyis a gép nem tud üzenni. Ezért az ON ERROR és POP utasításokat csak akkor írhatjuk be, ha már látjuk, hogy a program hibátlanul működik. A hibakezelés ON ERROR 0 utasítással kikapcsolható.

Függvények

INSTRING FN i (start, string A, string B)

A start címtől kezdve átvizsgálja A sztringet, hogy megtalálható-e benne B sztring. Értéke B sztring első karakterének címe, ha megvan, és 0, ha nem találta. Az A sztring hossza tetszőleges, a B sztringé max. 255 karakter. Így működik a szótár keresése.

**MEMORY$** FN m$ //

A memória egészét sztringként adja vissza. Szakaszolható is: MEMORY$ // (28700 TO). Most már nincs más dolgunk, mint hogy az INSTRING-gel keresgéljünk benne.

```
LET k$ = "fogad"
LET c = INSTRING (28700, MEMORY$
// [TO 55760], k$)
```

Ekkor c értéke a fogad f betűjének címe lesz.

**STRING$** FN s$ (szám, sztring)

Eredménye: a sztringnek számszori ismétlődése.

```
STRING$ (32, "x") = 32 db x
STRING$ (2, "ha") = haha
```

Jól használható adott memóriaterületek azonos karakterekkel való gyors feltöltésére. Például fájl törlése: STRING$ (3500, CHR$ 0) = 3500 db nulla. Ehhez viszont helyet kell biztosítani a RAMTOP alatt.

A szótárprogram a fenti utasítások és függvények segítségével működik, használatát a következő részben ismertetjük.

LITAUSZKY GYÖRGY

A Melbourne Draw

A már klasszikusnak számító ZX—Spectrum programok egyike a Melbourne House szoftvercég Melbourne Draw (M draw) nevű terméke, mely színes képek készítésére, átrajzolására, átszínezésére, feliratozására, javítására és UDG karakterek szerkesztésére alkalmas. Lehetőséget ad a meglévő képernyőfájlok betöltésére, módosítására, javítására, feliratozására, majd szalagra mentésére.

A Spectrum-kedvelők nagy része különböző módokon jut programjaihoz — jobb esetben az eredeti angol vagy más idegen nyelvű leírással együtt, de legtöbbször egymás közötti csere útján. Ilyenkor gondolkat okoz a program használata, mert próbálkozással legritkább esetben derül ki minden kezelési fogás és lehetőség. Ezen a problémán kívánok segíteni az M draw ismertetésével.

A program egy BASIC (1796 bájt) és egy gépi kódú (6600 bájt) részből áll. Közvetlenül a betöltés után a képernyőn az *1. ábra* szerinti alapválaszték (menü) jelenik meg, amelyből a kiválasztott feladatot a hozzá tartozó kezdőbetűvel lehet indítani. A jelölések: p — rajzolatszerkesztés; e — visszatérés BASIC-be; s — képkimentés szalagra; l — képbetöltés szalagról; v — képkimentés ellenőrzése; S — UDG betűk kimentése; L — UDG betűk betöltése; V — UDG betűk kimentésének ellenőrzése.

A P billentyű megnyomása után a képernyőn a kimentéskor használt papírszínnek megfelelő 32 × 22 karakter méretű üres mező jelenik meg, melynek közepén egy alig látható kurzor villog. A két alsó sorban a *2. ábra* szerinti felirat látható. Ez azt jelenti, hogy SCREEN MOD-ban, SKIP állásban vagyunk, a kurzor X és Y koordinátái 127 és 87, a P mellett a papírszín, az I mellett a tintaszín látszik, a nagy kocka pedig az „irányjelző", amely a későbbiekben ismertetett módon jelzi, hogy a képnek melyik része lett kinagyítva. A felirat SYMBOL SHIFT és 9 billentyűkkel a felső két sorba helyezhető, újabb megnyomásra visszaáll az alsó részre.

Képek, ábrák rajzolása

A G billentyű megnyomása után szürkefehér, karakter méretű sakktábla jelenik meg, ami a rajzoláshoz nyújt segítséget. A kurzort mozgató billentyűk az S betűt fogják körül (W — fel; X — le; A — balra; D — jobbra; Q — balra fel; Z — balra le; E — jobbra fel; C — jobbra le).

A SPACE billentyű lenyomására SKIF állapot lép fel. Ekkor a kurzor a mozgatógombokkal úgy mozgatható, hogy *nem húz vonalat*. Az ENTER billentyű lenyomására SET állapot lép fel, és a kurzor *vonalat húz*. Az O billentyű lenyomása után a már berajzolt vonalat törölni lehet. Az I billentyű lenyomása INVERT hatású, azaz, ahol vonal volt, ott töröl, ahol nem volt, ott vonalat húz.

Az M billentyű hatására a kép a kurzorállás közelében kinagyítódik, a billentyűt

```
Main Menu
<p> > edit picture
<e> > exit program
<s> > save picture to tape
<l> > load picture from tape
<v> > verify picture on tape
<S> > save UDG area to tape
<L> > load UDG area from tape
<V> > verify UDG area on tape

Press the key in the brackets
 for the desired command.
```

1. ábra

```
MODE SKIP.   SCREEN
X = 127      Y = 87
```

2. ábra

újra lenyomva, tovább nagyítódik. Ez szineskép állapotra is vonatkozik. A visszakicsinyítés az N billentyű egyszeri, illetve kétszeri lenyomásával történik.

A G billentyű újbóli lenyomásakor a kép papírszínű alapon tintaszínnel fog látszani. A papírszín SYMBOL SHIFT és számok billentyűkkel (0—7-ig), a tintaszín csak 0—7 billentyűkkel állítható.

A K billentyű hatására a rajzolat a kurzormozgató billentyűkkel 8 irányba mozgatható (SCROLL) — a színek nem!

A T billentyű lenyomására egy belül üres nyíl jelenik meg, és ettől a pozíciótól a nyíl irányába karaktereket (UDG-t is) lehet beírni. A beírandó betűnek megfelelő billentyű megnyomása nagybetűt, a CAPS SHIFT és betűbillentyű megnyomása kisbetűt eredményez. A CAPS SHIFT és SYMBOL SHIFT billentyűk egyidejű lenyomására az addig belül üres nyíl fekete színűre változik, és a CAPS SHIFT és 5; 6; 7; 8 billentyűkkel a nyíl más irányba fordítható. Ezután a nyíl belül újra üres lesz, és a feliratozás az új irányba folytatható. Ebből az üzemmódból való kilépés rövid BREAK-kel történik. A betűkre ékezet SET módban, a rajzolásra vonatkozó eljárás szerint készíthető.

A B billentyű után ugyancsak lenyomjuk a keret kívánt színének megfelelő billentyűt, és a keret azonnal erre a színre vált.

UDG karakterek szerkesztése

A G billentyűvel a hátteret szürke-fehér sakktáblaállásba kell hozni, majd a kurzort beállítjuk valamelyik háttérkocka *bal felső* sarkába, és a SKIP és SET állások használatával (SPACE és ENTER billentyűkkel) — kinagyítva könnyebb! — a kockába beírható a 8 × 8-as UDG karakter. Ez után az U billentyű lenyomására a gép megkérdezi: „Melyik betű legyen?" Beírjuk a kiválasztottat. Ekkor a gép erre definiálja, alulra pedig kiírja a DATA-t. Az így megszerkesztett karakterek szalagra kimenthetők és szalagról betölthetők.

Rajzok méretének változtatása

A SYMBOL SHIFT és 8 billentyűk egyidejű lenyomására MOD SCALE állapotba kerülünk, és egy választék (menü) jelenik meg a képernyőn.

Az E billentyű lenyomására a program visszatér MOD SCREEN állapotba. A D billentyű hatására a kép X—Y irányban

összébb megy, ismételt SYMBOL SHIFT és 8, valamint D lenyomására tovább zsugorodik a kép. Az I billentyűvel X—Y irányba növelhető a rajz, de *részekre szakad!*

Attributumok módosítása

A program az attributumok módosítására is alkalmas. A „H" billentyű lenyomásával kerülhetünk ebbe az állapotba; alul a felirat ekkor MOD SKIP ATTR vagy MOD SET ATTR lesz.

A SPACE billentyű hatására SKIP állapot lép fel; egy karakter méretű színes kurzor mozgatható a képen, de *nem hagy nyomot.* Az ENTER billentyű hatására a karakter méretű kurzor *nyomot hagy* a korábban beállított papír és tinta színeivel. A papír azonnal látszik, a tinta csak ott, ahol a fekete-fehér üzemmódban látható rajz van.

A SYMBOL SHIFT és V billentyűk egyidejű lenyomásával villogó karakter (FLASH), a SYMBOL SHIFT és B billentyűk egyidejű lenyomásával dupla fényes (BRIGHT) karakter állítható be. Kilépés ezek ismételt lenyomásával lehetséges. A SET és SKIP állapot itt is az ENTER és SPACE billentyűkkel állítható.

A K gombbal a színek (ATTR) mozgathatók (SCROLL) a rajzolat változatlanul hagyása mellett.

Minden zárt alakzat tintaszínnel kitölthető. A kitöltéshez a kurzort SKIP állapotban (a SPACE billentyű lenyomása után) be kell vinni a zárt alakzat belsejébe, majd SYMBOL SHIFT és F billentyűk egyidejű lenyomására az alakzat kitöltődik.

A SIMBOL SHIFT és L billentyűk egyidejű lenyomására a rajz áttükröződik, ismételt lenyomásra visszaáll az eredeti állapotba.

A rajz átszínezése, törlése

A SYMBOL SHIFT és R billentyűk egyidejű lenyomására az alábbi választék jelenik meg:

CLEAR PAPER (P)	— a papírszín módosítása
INK (I)	— a tintaszín módosítása
BOTH (B)	— mindkét szín módosítása
SCREEN (S)	— a rajz törlése
ALL (A)	— teljes törlés
(N)	— kilépés ebből az üzemmódból

A program BREAK-kel (CAPS SHIFT és SPACE) megszakítható, például kazettára való kimentés előtt, amikor a fő választékhoz akarunk visszatérni. Az M draw program BASIC-ből GO TO 3Ø-cal újraindítható.

A program gépi kódú része a tárban a 40960 memóriacímtől kezdődően helyezkedik el. A szalagra kimentett rajzolat, mivel ez a 32768-39680 memóriacímen helyezkedik el, más programban való felhasználás esetén betöltéskor csak akkor fog azonnal látszani, ha LOAD "név" CODE 16384,6912 vagy LOAD "név" SCREEN$ utasítással olvassuk be.

VADÁSZ LÁSZLÓ

ZX–SPECTRUM

Programgyorsító

A µMagazin legutóbbi számait olvasva több olyan gépi kódú programmal találkoztunk, amely a Spectrum képernyőjét kezeli. Mi két olyan, ebben a témában gyakran alkalmazható rutint mutatunk be, amelyek könnyebbé és gyorsabbá teszik a programozást. Véleményünk szerint rutinjaink a lehető leggyorsabbra és legrövidebbre sikerültek.

A memóriamutatók 000xH értéke mindkettő esetében természetesen fiktív, csak a programlépések relatív elhelyezkedésének bemutatására szolgál. Egyik szubrutin sem vizsgálja, hogy a kiszámított cím a képernyőre esik-e, ezt a felhasználónak kell figyelnie.

Az UP szubrutin

Kiszámítja a HL regiszterpár által megadott képernyőbájt feletti képbájt címét és betölti a HL regiszterpárba. A használt regiszterek: A és HL. Hossza: 15 bájt. V. i: 1. 46 T áll.; 2. 65 T áll.; 3. 26 T áll.

Bemenet: HL — kezdőcím. Kimenet: HL — HL_{bem} feletti cím.

A Spectrum képernyőtérképe:

4000	4001	...	401F
4100	4101	...	411F
.	.		
.	.		
4700	4701	...	471F
4020	4021	...	403F
.	.		
.	.		
4720	4721	...	473F
.	.		
.	.		
40E0	40E1	...	40FF
.	.		
.	.		
47E0	47E1	...	47FF
4800	4801	...	481F
.	.		
.	.		
4F00	4F01	...	4F1F
.	.		
.	.		
48F0	48E1	...	48FF
.	.		
.	.		
4FE0	4FE1	...	4FFF
5000	5001	...	501F
.	.		
.	.		
5700	5701	...	571F
.	.		
.	.		
50E0	50E1	...	50FF
.	.		
.	.		
57E0	57E1	...	57FF

A szubrutin alapesetei

1. A nyolckarakteres részek határánál (a képernyőharmadoknál)
2. Karakterek határánál, a fenti esetek kivételével
3. Egyéb esetben, azaz nem karakterek határátmeneténél

A teendők szétválasztása

Eset	H	L	Művelet
1.	xxxxx000	<32	H=H−1; L=L+224.
2.	xxxxx000	≥32	H=H+7; L=L+224.
3.	nem xxxxx000 alakú	bármi	H=H−1

Egy végrehajtott DEC H után

Eset	H	L	Művelet
1.	xxxxx111	<32	L=L+224.
2.	xxxxx111	≥32	H=H+8; L=L+224.
3.	nem xxxxx111 alakú	bármi	nincs

(x jelentése: a bit értéke akármi)

Memória		OP kód	Címke	Mnemonik
0000	:	7C	UP:	LD A,H
0001	:	25		DEC H
0002	:	E6 07		AND 7
0004	:	C0		RET NZ
0005	:	7D		LD A,L
0006	:	C6 E0		ADD 224
0008	:	6F		LD L,A
0009	:	DO		RET NC
000A	:	7C		LD A,H
000B	:	C6 08		ADD 8
000D	:	67		LD H,A
000E	:	C9		RET

Magyarázat

A H regiszter A-ba töltése és H csökkentése. Az eredeti H érték legalsó három bitjéből legalább 1 zérus: visszatérés (3. eset). Az L regiszter értékének A-ba töltése, az A növelése 224-gyel és az eredmény visszatöltése L-be. Ha L<32 volt: visszatérés (1. eset). A H regiszter értékének A-ba töltése, az A növelése 8-cal és az eredmény visszatöltése H-ba. Visszatérés a hívóhoz (2. eset).

A DOWN szubrutin

Kiszámítja a HL regiszterpár által megadott képernyőbájt alatti képbájt címét és betölti a HL regiszterpárba. A használt re-

utinok

giszterek: A és HL. Hossza 15 bájt. V. i.: 1. 46 T áll.; 2. 65 T áll.; 26 T. áll.
Bemenet: HL — kezdőcím. Kimenet: HL — HL_{bem} alatti cím.

A szubrutin alapesetei

1. A nyolckarakteres részek határánál (a képernyőharmadoknál).
2. Karakterek határánál, a fenti esetek kivételével.
3. Egyéb esetben, azaz nem karakterek határátmeneténél.

A teendők szétválasztása

Eset	H	L	Művelet
1.	xxxxx111	≥224	H=H+1; L=L−224.
2.	xxxxx111	<224	H=H−7; L=L−224.
3	nem xxxxx111 alakú	bármi	H=H+1.

Egy végrehajtott INC H után

Eset	H	L	Művelet
1.	xxxxx000	≥224	L=L−224.
2.	xxxxx000	<224	H=H−8; L=L−224.
3.	nem xxxxx000 alakú	bármi	nincs

(x jelentése: a bit értéke akármi)

Memória		OP kód	Címke	Mnemonik
0000	:	24	DOWN:	INC H
0001	:	7C		LD A,H
0002	:	E6 07		AND 7
0004	:	C0		RET NZ
0005	:	7D		LD A,L
0006	:	D6 E0		SUB 224
0008	:	6F		LD L,A
0009	:	D0		RET NC
000A	:	7C		LD A,H
000B	:	D6 08		SUB 8
000D	:	67		LD H,A
000E	:	C9		RET

Magyarázat

A H regiszter értékének növelése. H legalsó három bitjének vizsgálata; ha legalább egy zérustól különböző: visszatérés (3. eset). Az L regiszter értékének A-ba töltése, 224 kivonása A-ból és az eredmény visszatöltése L-be. Ha L≥0: visszatérés (1. eset). A H regiszter értékének A-ba töltése, A csökkentése 8-cal és az eredmény visszatöltése H-ba. Visszatérés a hívóhoz (2. eset).

CENTGRAF TAMÁS—
GYETVAI ÁRPÁD

256 × 192-es grafika

ZX–SPECTRUM

Sinclair Spectrum számítógépen, BASIC-ben dolgozva, a PLOT utasítással általában csak a felső 176 pontsort érhetjük el; csak óda rajzolhatunk, a fennmaradó, alsó 16 sorba nem. Az alábbi rutin segítségével a teljes, 6 kbájtos képernyőt telerajzolhatjuk.

```
PLOT LD BC,/COORDS/;  koordináták betöltése
     LD A,191;        belépünk a címszámító
     CALL 22ACH;      rutinba, de nem az elején
     JP ØD4DH;        befejezés a színek beállításával
```

Azok számára, akik nem annyira érteni, mint inkább használni akarják a rutint, közlöm a betöltőprogramot is, egy kis demonstrációs résszel. Ez utóbbi kirajzolja a képernyőre a lehetséges legnagyobb ellipszist, s egyben bemutatja a rutin használatát.

```
1Ø DATA 237,75,125,92,62,191,2Ø5,
172,34,2Ø5,236,34,195,77,13
2Ø CLEAR 65509: FOR X=65510 TO
   65524: READ A: POKE X,A: NEXT
   X
3Ø REM ---- DEMO ----
4Ø FOR Z=Ø TO 2*PI STEP .Ø1: LET
   Y=96+95*SIN Z: LET X=128+
+127*COS Z: POKE 23677,X: POKE
   23678,Y: RANDOMIZE USR 65510:
   NEXT Z
5Ø PAUSE Ø
```

Megjegyzés: az Y koordináta a rutin használatakor Ø és 191 közötti nagyságú lehet. 191-nél nagyobb vagy Ø-nál kisebb értékre a szokásos módon reagál a gép. Ø-nál kisebb értéknél már a POKE utasítással is gondjaink lesznek.

A képernyő invertálása

Sokszor lehet szükség valamilyen figyelemfelkeltő műveletre. Ha máshol nem, játékprogramokban biztosan elkél egy olyan rutin, mely a képernyőt invertálja, méghozzá igen nagy sebességgel.

A legegyszerűbb eljárás az lenne, ha a képernyő bittérkép-területét, annak minden egyes bájtját komplemensére fordítanánk. Kellemes megoldás, csak kissé lassú, még gépi kódban is. Ha azonban nem itt, hanem az attributumoknál ügyködünk, már felgyorsulnak az események.

Az alábbi program minden egyes karakterhelyen felcseréli a tinta és a papír színét.

```
INVERT LD HL,22528
       LD BC,7Ø4
CIKL   LD A,/HL/
       LD D,A
       AND 7
       SLA A
       SLA A
       SLA A
       LD E,A
       LD A,D
       AND 56
       SRL A
       SRL A
       SRL A
       OR E
       LD E,A
       LD A,D
       AND 192
       OR E
       LD /HL/,A
       INC HL
       DEC BC
       LD A,B
       OR C
       JR NZ,CIKL
       RET
```

A BASIC betöltő:

```
1Ø DATA 33,Ø,88,1,192,2,126,87,23Ø,7,2Ø3,
39,2Ø3,39,2Ø3,39
2Ø DATA 95,122,23Ø,56,2Ø3,63,2Ø3,63,2Ø3,
63,179,95,122,23Ø
3Ø DATA 192,179,119,35,11,12Ø,177,32,
223,2Ø1
4Ø CLEAR 65299: FOR X=653ØØ TO
   65339: READ A: POKE X,A:
   NEXT X
```

Látványosságáról, gyorsaságáról a következő sor hozzáírásával győződhetünk meg:

```
5Ø LIST: LIST: LIST: FOR X=Ø TO 4Ø:
   RANDOMIZE USR 653ØØ: PA
   USE 1: NEXT X
```

A program a tárban bárhová áthelyezhető.

Megjegyzés. Így ebben a formában csak a képernyő felső 22 karaktersora invertálódik. Ha ebbe az alsó kettőt is bele kívánjuk vonni, a 1Ø-es DATA-sorban javítsuk ki az ötödik számot 255-re.

JÁKSÓ LÁSZLÓ

Rajzoló

Ezzel a gépi kódú programmal a képernyőre lehet rajzolni, törölve rajzolni, le lehet törölni a képernyőt, a képernyő tartalmát elmenteni, az elmentett képet a képernyőre vinni, ott megnézni, majd ezután vissza lehet térni a BASIC-be. Az elraktározott kép láthatóvá tételéhez azt használom ki, hogy az OUT-nál a 0—63-mas portoknál a negyedik bit magasra állításával a képernyőt 8 kbájttal alacsonyabb címre lehet hozni.

A program csak A32-es gépen működik. Az assembler listában az aláhúzott bájtokhoz A48 esetében 64-et, A64 esetében 128-at kell hozzáadni.

A parancsok:

"A"	fel
"Y"	le
"<"	balra
"ú"	jobbra
"SPACE"	lenyomás esetén nem rajzol, hanem töröl
"K"	képáttöltés egy másik memóriaterületre (2. lista)
"E"	visszatöltés
"N"	az elraktározott kép elővétele
"R"	visszakapcsolás
"CLS"	képernyőtörlés
"B"	billentyűvel lehet a BASIC-be menni; ekkor hangjelzés után három funkció van:
"S"	kimenti a képernyő tartalmát
"V"	vége a programnak
"U"	visszatérés a rajzoláshoz.

HORVÁTH ISTVÁN

PRIMO

```
RAJZOLO

 1  4700   IN A,(35H)
 2  4702   AND 01H
 3  4704   JR Z,4709H   ; CLS LENYOMOTT
 4  4706   CALL 01C9H   ; KEP TORLES
 5  4709   IN A,(20H)
 6  470B   AND 01H
 7  470D   JR Z,4713H   ; N LENYOMOTT
 8  470F   LD A,88H
 9  4711   OUT (00H),A
10  4713   IN A,(14H)
11  4715   AND 01H
12  4717   JR Z,471DH   ; R LENYOMOTT
13  4719   LD A,88H
14  471B   OUT (00H),A
15  471D   IN A,(04H)
16  471F   AND 01H
17  4721   JR Z,4730H   ; E LENYOMOTT
18  4123   LD A,68H
19  4125   LD (47F5H),A
20  4728   LD A,48H
21  472A   LD (47F8H),A
22  472D   CALL 47F2H
23  4730   IN A,(28H)
24  4732   AND 01H
25  4734   JR Z,4743H   ; K LENYOMOTT
26  4736   LD A,48H
27  4738   LD (47F5H),A
28  473B   LD A,68H
29  473D   LD (47F8H),A
30  4740   CALL 47F2H
31  4743   IN A,(1AH)
32  4745   AND 01H
33  4747   JR Z,474AH   ; B LENYOMOTT
34  4749   RET
35  474A   LD BC,4F00H ; RAJZOLASI
36  474D   DEC BC      ; SEBESSEG
37  474E   LD A,B
38  474F   OR C
39  4750   JR NZ,474DH
```

```
40  4752   IN A,(33H)
41  4754   AND 01H
42  4756   JR Z,4759H   ; U LENYOMOTT
43  4758   INC DE
44  4759   IN A,(27H)
45  475B   AND 01H
46  475D   JR Z,4768H   ; < LENYOMOTT
47  475F   DEC DE
48  4760   IN A,(08H)
49  4762   AND 01H
50  4764   JR Z,4767H   ; V LENYOMOTT
51  4766   INC D
52  4767   IN A,(0EH)
53  4769   AND 01H
54  476B   JR Z,476EH   ; A LENYOMOTT
55  476D   DEC D
56  476E   IN A,(19H)
57  4770   AND 01H
58  4772   JR Z,4778H   ; SPACE LENYOMOTT
59  4774   CALL 3F18H   ; PONT TORLES
60  4777   NOP
61  4778   JP 4700H
62  477B   CALL 3FF8H   ; PONT KIJELOLES
63  477E   JP 4700H
```

1. lista

2. lista

```
1  47F2   PUSH DE
2  47F3   LD DE,6800H
3  47F6   LD HL,4800H
4  47F9   LD BC,1800H
5  47FC   LDIR
6  47FE   POP DE
7  47FF   RET
```


Szerkesztőségünk bővülő feladataihoz

munkatársakat keres

Jelentkezés

— telefonon: 154-090

— személyesen: Bp. II., Fő u. 68. IV. em. 452.

— írásban: 1027 Bp., Fő u. 68.

ADOK — VESZEK CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közületeknek gépelt soronként (60 karakter) 100,— Ft, magánszemélyeknek az első sor 50,— Ft, minden további sor 20,— Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

■ Játékprogramokat cserélek és a következő programokat: FORTH, LOGO.BIN, G-Pascal 2049, PROFI ASS 64, beszédgenerátoros és feltölthető szótár és nyelvteszt-program. Commodore 64 gépre, kazettás egységre valók. Cserébe játékprogramokat kérek, játéklistával együtt. Kovács Sándor, Bp. Füredi u. 64—66. VI. 75. 1144.

■ Commodere 64-re készült játékprogramokat cserélek. Kiss János, Hódmezővásárhely, Árpád u. 2. 6800.

■ C64-re keresek „Sz-mon" című programot! Csere! Pinczési István, Dunaújváros, Szorád út 46. 2400. Telefon: 171-87.

■ Commodore 64-re készült játékprogramokat cserélek. A válaszokat listával kérem. Czakó Zsolt, Lábatlan, Kun Béla tp. 5/3. 2541.

■ C64-es játékprogramokat cserélnék és keresem a TURBO COPY II-t. Németh Norbert, Szombathely, Kun Béla u. 47. III. 27. 9700.

■ Commodore VC—20-as programok eladása. Assembler, dissembler, karaktermódosító finomgrafika, zene stb. 100—300 Ft/db.. Kérjen tájékoztatót! Levélcím: Juhász György, Salgótarján, Pf. 157. 3100.

■ Repülést szimuláló programot keresek C16-ra (szalagra). Nagy Gyula, Kálvin u. 44. 5700.

■ ZX-Spectrumra játékprogramokat cserélek. Cím: Szentendre, Kossuth u. 12. 2000. Telefon: 06-26-11 648.

■ VC—20 számítógép, Commodore magnóval, dokumentációval 12 000 forintért eladó. Czene István, Abony, Szelei út 60. 2740.

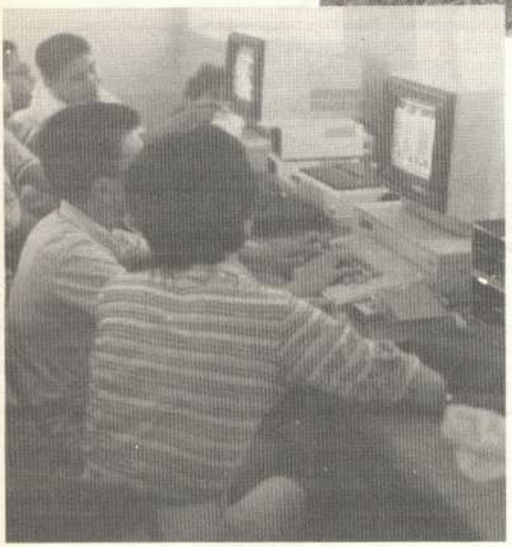
■ ZX-Spectrum 16 k-s személyi számítógép 8000 forintért eladó. Kovács Levente, Mezőberény, Árpád u. 35. 5650. Telefon: 66/51-159.

■ ZX—Spectrum+ számítógépemet eladnám vagy elcserélném videokimenettel, 200 felhasználói- és játékprogrammal, felhasználói kézikönyvekkel C64-re megállapodás szerint, *lehet hibás is.* Vásárolnék hibás C64-et és Commodore floppyt.
Kiss Tamás, Szeged, Ipoly sor 7/A. 6724. T.: 06-62/28-124

■ „C 64-hez játék és egyéb programokat cserélek.
VC 20-hoz játék eladó.
Cartridge
Micsik Zoltán, Szeged, 6720 Szent Miklós u. 3. 3/3.

Pagodák között

Számítástechnikai eszközöket árusító üzlet



Számítástechnikai oktatókabinet

Kina az ellentétek országa. Már érkezésünkkor, a vonat ablakából való első ismerkedésünk ezt mutatta: szemünk láttára nőttek a mongol határ környéki sárkunyhók előbb a több évezredes, monumentális Nagy Fallá, majd a még intenzíven épülő pekingi felhőkarcolókká. És magában Pekingben is, a kerékpárosok sokezres hada (a tízmilliós városban csaknem hatmillió kerékpár van) karikázik a széles utcákon a számítógépüzletek előtt. Az ezek melletti szűk közbe befordulva, a tenyérnyi lakásokból a lakók fele az utcára kiszorulva, a bejárat előtt — a keskeny sikátort félig eltorlaszolva — éppen ebédel.

Ezek az ellentétek azonban az óriási léptékű fejlődési ütem természetes velejárói. A gazdasági élet láthatóan pezsgő. Tág teret kaptak például a kisvállalkozások, a magánétkezdéktől kezdve a számítógépüzletekig. Az utóbbiakból az utcai sétán közben is sok tucatot láttunk, a távoli külvárosban is. Ez még magyar szemmel is feltűnt, nemhogy a többi kelet-európai országból érkezett kollégáknak.

Házi- és iskolaszámítógépek

A kirakatokban ritkán láttunk kis teljesítményű, háztartási tévéhez és magnóhoz csatlakoztatható mikroszámítógépeket. Közülük csak japán gyártmányú, MSX szabványú gépekkel találkoztunk az üzletben, 800 jüanos áron. (1 jüan = kb. 13 forint, az átlagfizetés csupán 80-100 jüan.)

Hivatalos források szerint Kínában jelenleg 23 vállalat állít elő háziszámítógépeket. Az 1985-ben gyártott mennyiség 150 ezer, 1986-ra pedig 200 ezer darabot terveznek. Jelenleg kétféle háziszámítógépet gyártanak. Áruk 1985-ben még 1000 jüan felett volt, most már a hivatalos források szerint 300 jüan körülire csökkent.

Az iskolaszámítógép-program a szakosított tantervű iskolákkal indult; ma már valamennyi ilyen iskolában van legalább egy mikroszámítógép. A nagyvárosok iskoláiban is gyorsan terjedtek a mikrogépek. Például a pekingi iskolák 80-90 százalékának van már iskolaszámítógépe. Vidéken, a kisebb településeken azonban még ritka jelenség a mikroszámítógép.

Professzionális mikroszámítógépek

Az utcákon látott üzletek többsége csak ezzel a kategóriával foglalkozik. Szemmel láthatóan nem csupán értékesítésről van szó, hanem a komplex szolgáltatások felé törekednek. Az üzletekben ugyanis több helyen láttunk hardverjavítást, szakirodalom- és szoftverértékesítést (az utóbbit természetesen bemutatóval illusztrálva), és a feliratok szerint szoftverfejlesztési tevékenységre való vállalkozási készséget is.

A kisebb, a háziszámítógéppel való átmenetet képező, professzionális gépnek tekinthető Apple II eredeti gép ára 6500 jüan, de tízezres sorozatnagysággal már kínai gyártása is beindult, Zijin II. néven. Sok iskola is kapott már ebből a típusból.

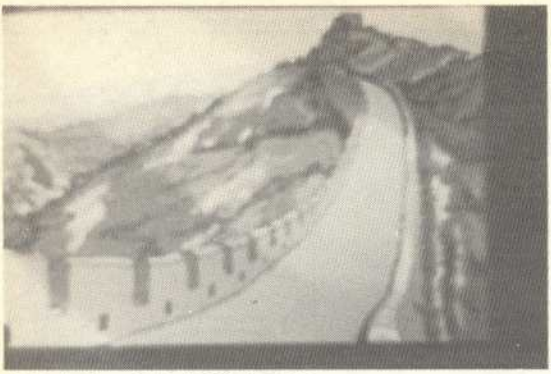
A számítástechnikai szaküzletek többsége szinte kizárólag az IBM PC-vel kompatibilis gépek forgalmazásával foglalkozik. Elsősorban a kínai gyártmányú Nagy Fal típust láthattuk a kirakatokban. Ára 520 kbájtos operatív tárral, egy hajlékonylemezes tárolóval 9500 jüan, Winchester-tárral és 2 db hajlékonylemezes tárolóval pedig 26-30 ezer jüan. A bőséges kínálat annak az eredménye, hogy e típust jelenleg már évi tízezres nagyságrendben gyártják, és a tervek szerint a sorozatnagyságot 1988-ra 100 ezerre kívánják bővíteni.

A hazai gyártású mikrogépek száma 1985 végén Kínában meghaladta a 200 ezer darabot. Ezenkívül több tízezer, elsősorban japán és hongkongi eredetű külföldi mikroszámítógépet értékesítettek már az országban.

Nagy teljesítményű számítógépek

1984-ben fejezte be az Észak-Kínai Számítástechnikai Kutatóintézet a 32 bites, 2780 típusú miniszámítógép fejlesztését, melynek bevizsgálása 1985 közepére ért véget. Ez a típus, mint megnevezése is sejteti, a VAX 11/780-nal kompatibilis. Műveletvégzési sebessége egymillió művelet másodpercenként.

Egy évvel korábban, 1983 májusában Dél-Kínában, a



A kínai „Nagy Fal" számítógépes képe

Changsha-i Honvédelmi Technológiai és Tudományegyetemen befejezték a Yinhe (tejút) nevű nagyszámítógép fejlesztését. Ezt fél évig vizsgáztatta egy állami bizottság, és utána megadták a sorozatgyártási engedélyt. 1984-ben már több mint 60 készült belőle. A Yinhe 100 millió műveletet végez másodpercenként. Meg kell jegyezni, hogy az európai szocialista országokban még nem készült ekkora teljesítményű gép. A KGST-országokban gyártott nagyszámítógépek közül ma a legnagyobb teljesítményt a szovjet ESZ-1045 képviseli, három nagyságrenddel kisebb, 880 000 művelet/másodperces sebességgel.

A Yinhe hatalmas teljesítményét például nagyon jól ki tudják használni a szeizmikus kutatásokat végző intézetben. Ennek feladata Kína legveszélyeztetettebb területein a földrengéseket kiváltó tényezők vizsgálata előrejelzési célból. A Yinhe alkalmazásba vétele új távlatokat nyitott meg a földrengések prognosztizálása terén.

Robotok

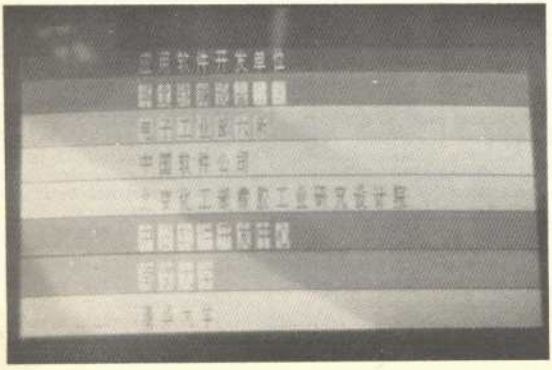
Az utóbbi években egyre több robot jelenik meg Kínában. Jelenleg körülbelül 100 robot és kétezer különféle tagolt kar használatos. A jelenlegiek többnyire első generációs robotok, melyeknek viszonylag egyszerű a működésük. A második generációs robotok még kísérleti fázisban vannak, nem használják őket a termelésben.

A fejlesztések során prioritást kapnak az olyan robotok, amelyek rossz körülmények között: magas hőmérsékleten, poros és zajos helyen, erős radioaktív sugárzásban, intenzív szeizmikus mozgások mellett is képesek dolgozni.

Az utóbbi két évben Kína számítástechnikai ipara egyre dinamikusabban fejlődik. Ebben az iparágban jelenleg több mint 90 ezer fő dolgozik, és nyolc kutatóintézet, 111 üzem, 40 szolgáltató egység működik. A számítógépgyártás kapacitásának bővülése várhatóan maga után vonja az alkalmazási kultúra, az alkalmazások színvonalának emelkedését is.

DR. BROCZKÓ PÉTER

Kínai karakterek a képernyőn



A lemezmeghajtó egységszámának beállítása

A Commodore lemezmeghajtó egységeknek 8-as a gyári egységszámuk, de beépített lehetőség van az egységszám hardveres — áramköri — beállítására is. A beállítás módját a gépkönyvek többnyire tartalmazzák, csupán a német vagy angol gépkönyv mint nyelvi akadály jelent nehézséget. A hardveres egységszám-változtatás maga után vonja a garancia elvesztését is, ezért csakis garancián túl ajánlható.

E módszerrel az egységszám 8-9-10-11-re állítható be, míg a szoftveres átszámozással 8—15 között választhatunk. Az egységszámot a meghajtóegység 6522 vagy 65 C 22 integrált áramköre két kivezetésének földre — testpontra — való kötésének variációi adják; 8-at mindkét kivezetés földre kötése, 11-et pedig mindkettő felszabadítása jelenti. A közbeesők a két kivezetés felváltott 0—1, illetve 1—0 változatával érhetők el.

A gyakorlatban a lemezegység — VC—1541 vagy VC—1570-es meghajtót feltételezve — felső burkolatát kell leemelnünk az alsó rész négy sarkán hozzáférhető csavar kivételével. Ekkor elénk tárul a fémvázra erősített tápegység és az elektronikát tartalmazó áramköri lap, mely alatt a meghajtó mechanika rejtőzik a hozzá tartozó szabályozó áramkörökkel. A fémvázat további hat csavar rögzíti az alsó műanyag burkolathoz, melyek kicsavarása és a zöld LED vezetékének a panelról való oldása után a váz az egész egységgel, a lemezbevezető műanyag burkolattal együtt kiemelhető.

A LED-ek, az elektronika és mechanika közötti elektromos kapcsolatot a panelre forrasztott tüskesorok és a rájuk illeszkedő, gyengéd húzással eltávolítható csatlakozósávok jelentik. További bontás a leírt munkához nem szükséges, de célszerű a csatlakozók széthúzása előtt helyzetüket, sorrendjüket feljegyezni, a csatlakozókat és a panelt szín- és helyzetjelöléssel összekapcsolni.

Előre nézni, ráérezni, csinálni

Pécsett szerencsére sok a vállalkozó szellemű ember. Iskolakísérlet? Legyen! És lett belőle olyan intézményrendszer, amit ma az Általános Nevelési Központok közé elsőként sorolhatnak. 3-3 bölcsődéje, óvodája, általános iskolája, valamint művelődési háza, könyvtára, sportkombinátja, diákotthona mellé az 1986-os tanévtől még gimnáziuma is belép.

Tudtuk, hogy a mi nagy lehetőségünk az, hogy integrálhatjuk a nevelési, a tanítási és a közművelődési folyamatokat — és erre gondoltunk, amikor még 1984 szeptemberében beindítottuk az általános iskolások részére a fakultatív számítástechnikai oktatást. Hamar jött a feladat: meg kell szervezni a terem kihasználását úgy, hogy például egy hatodikos gyerek is ismerkedhessen a gépekkel, és a felnőtteknek is legyen módjuk tanulni.

Ugyanakkor tudomásomra jutott, hogy létezik egy társaság, melynek tagjai — hontalanként e hobbijukban — egyikük lakását egy-egy estére megszállják, és csak az érdekli őket, hogy miképpen működnek saját és kölcsönkért gépeik.

Hamar összefonódott a két szál, és megalakult az AGORA Computer CLUB, az Apáczai Nevelési Központ biztosította gépteremben, teljes eszköz-

Téves csatlakoztatás katasztrófát okozhat!

A lemezből készült fémváz hátsó síkján a hálózati biztosíték foglalata mellett egy viszonylag nagy méretű négyszögletes kivágás van, melyet a műanyag házzal célszerű összejelölni, mert az itt elhelyezendő kapcsoló részére a ház megfelelő helyén a szükséges nyílást ki kell majd vágnunk.

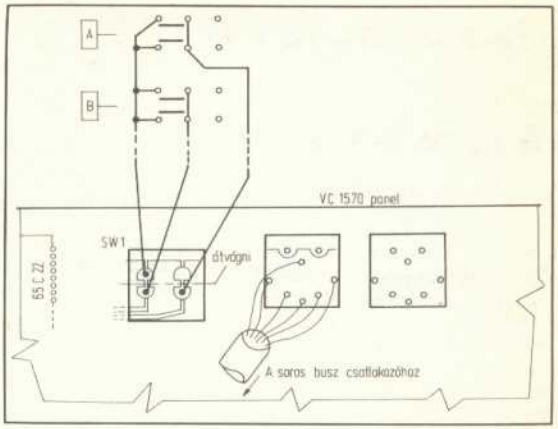
Kapcsolókénti alkalmas a szaküzletekben kb. 20 forintért kapható ISOSTAT, mely több kivitelben is készül. Céljainknak az felel meg legjobban, amely egymás mellett két darab, egyenként 2×3 kivezetést tartalmaz, és a két kapcsoló egymástól függetlenül benyomott, illetve kiugratott helyzetében rögzíthető. Ha ilyet nem kapunk, kis munkával — mivel ezek mechanikája csaknem azonos — egymás mellé szerelhetünk két így működő kapcsolót.

A kapcsolósor fémhídjának végén egymenetes furat szolgál rögzítésre, így a készülék fémvázán a házfedelek illeszkedési vonalában az ezeknek megfelelő 0,3 mm-es furatot elkészítjük. Fúráskor óvakodjunk az erőteljes rázástól és vigyázzunk, nehogy a fémszemcsék az elektronikára vagy a mechanikára szóródjanak!

A kapcsolók két-két áramkörét párhuzamosan kössük be úgy, hogy a felerősítés felé eső kivezetéseket válasszuk testpontnak, a középsőket pedig a váltandó „meleg"-pontnak. Ez a bekötés a kapcsolók kiugratott helyzetében jelenti majd az alaphelyzetet, azaz a 8-as egységszámot, amit feltehetően leggyakrabban fogunk használni, viszont a rugók így nyugalmi helyzetben lesznek. A szükséges vezetékhosszat ezután kössük a kapcsolókivezetésekre, tehát egy test, 2-3 melegvezeték, majd rögzítsük a kapcsolót a fémvázra.

A kapcsolók nyomógombjának megfelelően a szükséges nyílást fúrjuk, illetve vágjuk ki a készülékházon is. Ezután a váz visszaszerelhető az alsó dobozfedélbe.

Az elektronika paneljén keressük meg a fehér festékkel négyszög alakban körülhatárolt, „SW 1" feliratú áramköri részt. A VC—1541-es meghajtó esetében ez a panel kb. közepén, a VC—1570-esnél pedig a panel jobb oldali élének kb. közepén található (lásd a *rajzot*). A festékkereten belül két, egymás mellett elhelyezkedő, félkörökből álló fóliarész van. A félkörök átmérői



egymástól kb. 1 mm-es távolságban vannak, és középvonalukban egy-egy vékony fóliacsík köti össze őket. A félkörök közül egy-egy a panel szélén futó földpotenciállal, másik felük a 6522-es IC lábaihoz vezető fóliacsíkkal áll összeköttetésben. Ez utóbbiakhoz forrasszuk a két melegvezetéket, az előbbihez pedig a kapcsolók közös pontját.

A kapcsolókat kiugratott helyzetben hagyva, vágjuk át éles pengéjű késsel a félkörök átmérői között húzódó fóliahida(ka)t, és próbáljuk ki, jó munkát végeztünk-e.

A kapcsolók átváltását csak a készülék kikapcsolt állapotában végezzük, annál is inkább, mivel a 8-asként megszólított egység a továbbiakban akkor is a 8-asra „hallgat", ha netán közben a kapcsolókat átállítottuk más számra.

A kapcsoló fölé vagy mellé csinos piktogramot helyezünk el, hogy rajtunk kívül más is tudja, milyen kapcsolóállás milyen számot jelent.

LUKÁCS ATTILA

parkjának használatával. Módunk van egymás tapasztalatainak, programjainak megismerésére és cseréjére: mit kívánhat még egy megszállott mikrogépes?

Sokat. És igaza van, ha egyre igényesebb! Márpedig a tagok egymásra hatása eddig leginkább éppen abban nyilvánult meg, hogy közösen keresték egymás számára a problémák megoldását; így a legjobb kristályosodott ki. Ezt nevezik „brain storming"-nak, ha nem tévedek. Aztán itt van a gépek kérdése. Egyik fő törekvésünk, hogy különböző géptípusokat ismerjünk meg. Összehasonlításokat teszünk, értékelünk (kinek mire kellene és javasolható-e stb.). Volt már itt ORIC, Atari, mindenféle Sinclair és Commodore, VIDEOTON TVC, Primo, legújabban még Dragon is.

Hamis dolog lenne csak az AGORA CC-ről beszélnem, holott ez a terem főleg a gyerekekért van. 120 tanuló kap itt rendszeres fakultatív oktatást, rajtuk kívül még 40-50 más korú jön el többé-kevésbé rendszeresen a szerda és csütörtök esti nyílt klubokra. Sok ez vagy kevés? Nos, a feltétel nélküli lelkesedés kora — legalábbis nálunk — lejárt. Az 1600 felső tagozatos diákból így ez a létszám nemhogy nem rossz, hanem fölöttébb érdekes is! Nem vehető csodának, hogy az Országos Pedagógiai Intézet egyik kutatási témája is az itt folyó munkához kapcsolódik, amihez még sok C16-os géppel támogatást is nyújtott.

A készülékek üzemeltetésében sok tapasztalatunk gyűlt már össze. Sok a hibás C16-os gép: nem kezelik a soros buszon levő lemezegységet, printert, a gyári biztosítékok rendre megszakadnak stb. De próbált-e valaki 14 gépet egy teremben egyszerre működtetni? Interferenciás zavarok nélkül biztos nem sikerülhet. A megoldás: alapsávi videojeleket kibocsátanak a gépek, csak a televíziókat kell átalakítani. Az eredmény megéri a fáradságot!

Az eszközök vásárlása persze nekünk sem könnyű. A számítástechnika bevezetése a rendszerbe kezdettől fogva az oktatástechnikai osztály feladata volt, csakhogy erre pénzt külön nem kaptunk. Sokat kellett várnunk az első ZX81-es után az első VC20-ra, majd sok huzavona előzte meg a C64-es rendszer megvételét is. Amikor viszont a szaktanterem elkészült, már hozták a látogatókat ide is, mint ahogy a szomszédos televízióstúdióba.

Végül egy csemege. Egy hetedikes gyerek találmánya egy VC20-as gépre írt program, amelyet közreadok:

```
1Ø PRINT "♥ I LOVE YOU BASIC"
2Ø GOTO 1Ø
```

Kérdésem: miért „szemérmes" a VC20?

MAKÁNY GYÖRGY

Megalakult a Békásmegyeri Sinclair Club (B.S.C.)

Címe: Budapest III., Vendel u. 5. Telefon: 802-509 (Kovács Miklós)
Az üzemeltetés magánjellegű.
A klub vezetői:
Willner H. Gábor szekcióvezető
Horváth Béla szakkörvezető
Mindketten elsős gimnazisták.
Csoportos oktatás
minden szombaton 9—12 óráig
hétfőn 18.30—20 óráig
A tagsági díj
gyerekeknek havi 30 Ft
felnőtteknek havi 70 Ft

A géppark: 2 db ZX-Spectrum, 1 db VC—20 számítógép. A gépek a tagok tulajdonában vannak, és csak a szakköri időre engedik át azokat.

A szakkörben eddig csak Spectrum Basic-et oktattunk: 18 szakköri foglalkozáson alap BASIC-tanfolyamot tartottunk, a végén pedig saját készítésű ellenőrző teszttel vizsgázhattunk. Az eredménytől függően léphetnek a tagok a következő turnusra: 20 foglalkozáson keresztül a Spectrum belső felépítését ismertetjük, utána teszt.

A közeljövőben egy C-64 turnust is indítunk.

Szeretnénk kapcsolatot teremteni más klubokkal.

Willner H. Gábor — Horváth Béla

COMMODORE 64

A lemezkapacitás növelése

A közelmúltban egy Commodore 64 megbízás kapcsán szembekerültem a 1541-es floppy meglehetősen korlátozott tárolókapacitásával. A kezelendő adatmennyiség még olyan volt, hogy kísértést éreztem valami trükk alkalmazására, ami mégis megoldhatóvá teszi a szóban forgó feladatot.

Emlékeztem rá, hogy a Mikroszámítógép Magazinban láttam erre vonatkozóan egy cikket. Meg is találtam az 1985/3. számban. Problémám megoldásához figyelemre méltó ötletet adott. A közölt két megoldás közül az első — a 121-es számrendszerbe való konvertálás — látszott rugalmasabban alkalmazhatónak az én esetemben.

Első megközelítésben a konverziós rutint változtatás nélkül beépítettem a programomba. A működés során, az adatbeviteli fázisban, amiben 10-es számrendszerből 121-es számrendszerbe konvertáltam, nem is volt probléma. A lekérdezések, az adatfájlok rekordjainak feldolgozása során azonban, amikor az egyes mezőket 121-es számrendszerből 10-es számrendszerbe kellett egymás után, sorozatosan és tömegesen konvertálni, már jelentős időproblémák adódtak. Az adatbevitelnél ez a lassúság nem tűnt fel, mert elrejthető volt az egyes adatok bevitele közé, és egyébként sem okozhatott „feszültséget" a manuális tevékenységek környezetében. A visszaalakítás lassúsága azonban veszélyeztette a probléma jó gyakorlati megoldását, és erősen lehűtötte első örömömet.

A megbízást már elfogadtam, így nem volt más kiút, mint valahogy növelni a reményteljesebbnek hitt konverziós rutinok sebességét. Az elemzés során — amikor a közölt konverziós rutinokat elhelyeztem egy kis programkörnyezetben, ami adatokat kér be és kiírja a konvertálásra fordított időket — kiderült, hogy az eredeti megoldás a problémás visszaalakításnál 6-7 pozíciós számok esetén ~1 s-os idővel fut.

Ezután megnéztem, hogy a 121-es számrendszer karakterei (alaki értékei), illetve ezek sorrendje milyen összefüggésben van az ASCII kódtáblázattal? Kiderült, hogy lényegében semmi összefüggés nincs. Itt a sorrendet egyedül csak a billentyűzeten való elhelyezkedés sorrendje motiválta (?).

Átalakítottam tehát a konverziós rutinokat úgy, hogy az ASCII kódtáblázatból kiválasztottam 126 egymást követő (két szakaszban, *1. program*) értéket, és az ASCII kód alapján egy egyszerű számítással határoztam meg egy adott pozíció értékét. Így már nem kellett a sztringeket külön letárolni és sztringműveletek sorozatát elvégezni. Ezzel a megoldással a sebesség kb. 10-szeresére nőtt.

6-7 pozíciós decimális számoknál 0,1 s körüli idők adódtak, ami már a gyakorlati alkalmazás során is elfogadható futási időket eredményezett.

Elkészítettem a 126-os—10-es konverziót gépi kódban is. A rutin SYS 49152, BB$,SZ utasítással hívható, ahol a BBS-be kell elhelyezni a 126-os számrendszerbeli formát, és az SZ változóban kapjuk vissza a decimális megfelelőt. Így további sebességnövekedést értem el. Ez már az eredeti programhoz képes 30-40-szer volt gyorsabb (!).

A *2. program* a gépi kódú rutin BASIC betöltési formáját mutatja.

Természetesen ezeket a programokat is lehetne tovább tökéletesíteni.

Az elvileg lehetséges kódkészletből — a cikk által sugallt óvatosságból — csak egy részt használtam fel; ki lehetne próbálni a 0—255-ig terjedő egész tartományt (256-os számrendszer), mert az csak vizuális megjelenítés esetén okozhat problémát, de erre a tényleges működés során nincs szükség.

Gépi kódú megoldás esetén a jelenlegi számítási eljárás helyett a tömörített decimális ábrázolási formát feltehetően egyszerűbben lehetne megvalósítani.

MAGAS LÁSZLÓ

```
265 REM ************************************
270 REM ***KONVERTALAS 10-ESBOL 126-OSBA ***
280 REM ***HIVAS:GOSUB400                  ***
290 REM ***S/:DECIMALIS SZAM:BB$:ERETIMENY***
295 REM ************************************
300 A=SZ:BB$=""
310 E=INT(A/126):M=A-E*126
320 M=M+33:IFM>127THENM=M+33
330 BB$=CHR$(M)+BB$
340 IF E<>0 THEN A=E:GOTO310
350 RETURN
355 REM ************************************************
360 REM ***KONVERTALAS 126-OSBOL 10-ESBE            ***
370 REM ***HIVAS:GOSUB400                           ***
380 REM ***BB$:126-OS RENDSZERU SZAM,SZ:ERETIMENY    ***
390 REM ***H(0)=1:H(1)=126:H(2)=126*126:H(3)=H(2)*126***
395 REM ************************************************
400 SZ=0:N=LEN(BB$)
410 FOR I=1 TO N
420 C$=MID$(BB$,I,1)
430 K=ASC(C$):IFK>127THENK=K-33
440 K=K-33
450 SZ=SZ+H(N-1)*K
460 NEXTI
470 RETURN
```

1. program

2. program

```
500 DATA 32,115,0,32,139,176,133,167,132,168,32,115,0,32,139,176,133,176,132,177
510 DATA 168,0,177,167,170,32,94,192,168,32,162,179,166,176,164,177,32,215,187
520 DATA 168,0,177,167,170,202,240,46,32,122,192,32,147,192,168,0,177,167,170
530 DATA 202,202,240,31,32,122,192,32,132,192,32,147,192,168,0,177,167,170,202
540 DATA 202,202,240,12,32,122,192,32,132,192,32,132,192,32,147,192,96,160,1,177
550 DATA 167,133,164,168,2,177,167,133,165,138,168,136,177,164,201,128,144,3,56
560 DATA 233,33,56,233,33,96,32,94,192,168,32,162,179,32,15,188,168,126,32,162
570 DATA 179,32,168,192,32,48,186,32,15,188,96,165,176,164,177,32,162,187,32,168
580 DATA 192,32,186,184,166,176,164,177,32,215,187,96,165,102,69,110,133,111,165
590 DATA 97,96
600 S=0:FOR I=49152TO49328:READA:S=S+A:POKEI,A:NEXTI
610 IFS<>22599THENPRINT"HIBA A DATA SOROKBAN":STOP
620 PRINT"RENDBEN!"
```

ZX-SPECTRUM

Szövegkereső

Ez a rutin a K$ változóban lévő szöveget keresi meg a memóriában. Ha megtalálta, akkor az OK, 0:0 üzenettel leáll, és a megtalálási címet 23 550—23 551 címre teszi. A megtalálás helye:

PEEK 23 550+256*PEEK 23 551

Amennyiben nem talált ilyen sztringet, a „8 END OF FILE" üzenetet írja ki. Ha a K$ változó nem létezik, akkor a „2 VARIABLE NOT FOUND" üzenetet kapjuk.

A gépi kódú rutin a memória bármely részére betölthető; a javasolt hely valamely REM sor. Hossza 124 bájt, a keresést a RAMTOP-tól kezdi, és az UDG-nál fejezi be.

SZILÁGYI GYÖRGY

```
        LD   HL,(23627)
ISM     LD   A,(HL)
        CP   128
        JR   Z,RET
        AND  224
        CP   64
        JR   Z,MEGVAN
NEMK$   CP   96
        JR   NZ,NEMSZM
PLUSZ   LD   DE,6
        ADD  HL,DE
        JR   ISM
NEMSZM  CP   160
        JR   NZ,NEMHN
UTOLSO  INC  HL
        BIT  7,(HL)
        JR   Z,UTOLSO
        JR   PLUSZ
NEMHN   CP   224
        JR   NZ,NORMAL
        LD   DE,19
        ADD  HL,DE
        JR   ISM
NORMAL  INC  HL
        LD   E,(HL)
        INC  HL
        LD   D,(HL)
        INC  HL
        ADD  HL,DE
        JR   ISM
MEGVAN  LD   A,(HL)
        AND  31
        CP   11
JR  NZ,NORMAL
        INC  HL
        LD C,(HL)
        INC  HL
        LD B,(HL)
        INC  HL
        PUSH HL
        POP  DE
        PUSH DE
        POP  IX
RAMTOP  LD   HL,(23730)
KULSO   PUSH BC
BELSO   LD   A,(DE)
        CP   (HL)
        JR   NZ,NEMAZ
        INC  HL
        INC  DE
        DEC  BC
        LD   A,B
        OR   C
        JR   NZ,BELSO
        POP  BC
        JR   EZAZ
NEMAZ   PUSH IX
        POP  DE
        INC  HL
        PUSH HL
UDG     LD   BC,(23675)
        SBC  HL,BC
        JR   NC,NINCS
        POP  HL
        POP  BC
        JR   KULSO
NINCS   LD   (IY),7
        POP  HL
        POP  BC
        RET
EZAZ    SBC  HL,BC
        LD   (23550),HL
        RET
RET     LD   (IY),1
        RET
```

Amerikai előzetes

Júliusban három hetet töltöttem az Egyesült Államokban. Az út célja az általunk kifejlesztett különleges botkormányok bemutatása volt. Ezek közül az egyik az eddig botkormányt használni nem tudó óvodáskorúak számára készült, itthon az SzmSzm kiállításon bemutatott típus volt, melyről azt is megállapították, hogy egészségkárosultak gyógyításában is jól hasznosítható lenne. A másik típus a tíz év körülieket bűvölte el. A botkormányok bemutatása mellett klubokat is felkerestem, és kapcsolatot vettem fel néhány újabbal.

Mindezek miatt az út rohanás volt. Erre jellemző, hogy 20 nap alatt 21 alkalommal repültem! Ennek csak részben volt a program zsúfoltsága az oka. A másik okot a Northwest Orient légitársaság szenzációsan kedvező feltételű bérlete szolgáltatta. Ezzel a bérlettel 45 napig akárhányszor, akárhova lehetett utazni az Egyesült Államokon belül e légitársaság járatain, 450 dollárért. Az árra jellemző adat, hogy ha 45 napig továbbra is napi egy utat tehettem volna meg, úgy az ár utanként 10 dollár lett volna, ami kb. öt szendvics vagy két repülőtéri buszjegy ára, és természetesen magába foglalja a repülőgépen kapott ételeket és italokat is! Persze ez csábít sok utazásra, de a sok utazás oka részben az is volt, hogy a légitársaság legtöbb járata Minneapolisból indul; így én ötször voltam ebben a városban, de ebből négyszer csak a repülőtéren, mint ott átszálló utas.

Igyekeztem persze minél többet látni, amihez csak az egyik lehetőség volt a repülés. Volt olyan napom, amikor 800 kilométert autóztunk egyetlen városon belül, végiglátogatva jónéhány klubtagot és azokat a boltokat, ahol a klubtagok vásárolni szoktak. Meg kell jegyeznem azt is, hogy a város két vége közötti távolság kb. annyi volt, mint Budapesttől Székesfehérvárig. Amikor erre lehetőségem volt, gyalogoltam. Ez ott, részben a távolságok miatt, teljesen szokatlan. Így lehet azonban csak ott megállni, ahol akarok, abba a boltba bemenni, ahova akarok. Na nem vásárolni, mert ehhez pénz is kell, de meg lehet tudni, hogy mi mennyiért kapható, ki lehet próbálni a gépeket, programokat a jövőbeni vásárlás reményében.

Az úton szerzett ismeretek feldolgozása még tart. Itt most jórészt csak azt említem meg, hogy miről fogok írni: egy újfajta üzletrendszerről (afféle mindent egy helyen a számítástechnikában), az istenített-kételkedéssel fogadott, egykártyán-megvalósítottmerevlemezegységről, ismertetni fogok különböző programokat (Commodore 64-re, IBM PC kompatibilis gépekre, TRS 80 Co-Co-ra írtakat), az oktatással kapcsolatban megismerteket (néhány oktatószoftvert gyártó cég tapasztalatairól, egy főiskolán tett látogatásról). Közölni fogom a szoftvergyártás egy vezető szakemberének kizárólag számunkra adott interjúját, egy neves oktatási szakember cikkét. És természetesen írni fogok arról, hogy mi újat láttam a klubokban.

Említettem, hogy hányszor töltöttem el órákat az egyik repülőtéren. Igyekeztem ezt az időt is kihasználni, és mert a repülőtéren volt egy játékterem, egyik alkalommal ezt tanulmányoztam. Mielőtt még valaki azt hinné, hogy játékszenvedélyem kielégítése volt célom, megmondom, hogy egyetlenegy játékkal sem játszottam, viszont figyeltem a

játékokat és a játékosokat. A játékteremben 33 játékautomata vola. Mindegyiket a nagyméretű képernyő, a Commodore 64—Atari 800 szintű grafikánál nagyobb felbontású, szebb színű kép jellemezte. Kevés japán kivételtől eltekintve, a különböző automaták Atari gyártmányúak voltak.

A *céllövöldénél* puskával, pisztollyal lehetett a képernyőn megjelenő dolgokat lelőni. A jutalom játékidő volt. A *közlekedési* játékok között az ismert autóversenyen kívül volt olyan, amelynél egy motorkerékpárt kellett utcákon végigvezetni. A *lövöldözős játékoknál* különböző, visszalőni tudó objektumokat kellett lelőni, anélkül, hogy minket eltalálnának. A *harci játékoknál* a lövöldözésen kívül verekedni és akadályt mászni kellett. *Motorkerékpárvezetés:* egy valóságos motorkerékpár modellre ráülve kellett vezetni, a képernyőn látszott az útvonal és a játékos. Voltak még *félkarú banditák* és *golfjáték* is.

A játékterem iránt ottlétemkor meglehetősen csekély érdeklődés nyilvánult meg. Néhány bámészkodón kívül mindössze 17 játékos volt másfél óra alatt. Szerencsémre ketten végigjátszották az összes játékot. A legtöbben a lövöldözős-verekedős automatákkal játszottak.

Mivel itthon továbbra is vita folyik a milyen-számítógépet-az-iskolákba témakörben, talán érdemes megismerni, amit az ottani helyzetről az IBM több oktatási tanácsadójától hallottam, és amit mindkét oktatási szoftvert gyártó cégnél megerősítettek. Ez egy statisztika volt az iskolákban használt gépekről (a felsőoktatás kivételével). Eszerint a géppark mintegy 60 százaléka Apple-gyártmány (csaknem teljesen Apple II), 10—10 százaléka Commodore (szinte kizárólag 64, újabban 128 és elvétve még néhány VIC—20), Tandy (szinte kizárólag TRS 80 Color), IBM (szinte kizárólag PC vagy XT) gyártmány, 5 százalék Atari és a maradékon több cég osztozik. Az IBM igyekszik ezen a helyzeten változtatni, de ez valószínűleg nehezen fog sikerülni, mert a többiek — ha kevesebbet tudó gépekkel is, de — alacsonyabb árakkal dolgoznak, és az iskoláknak ha nem is annyira, de ott is meg kell gondolniuk, hogy mire és mennyit költsenek.

Az oktatószoftverek a géppark elosztásának megfelelő képet mutatnak. Apple II-re szinte minden van, míg a többiek jóreszt az ezekről átírt programokat használják. Az IBM ezen kíván változtatni, és ezzel kívánja sikerét biztosítani.

Végül ismertetem több ottani klub javaslatát. *Létre kellene hozni a klubok nemzetközi szervezetét, amely lapo(ka)t adna ki, időnként kiállításokat, vásárokat, konferenciákat tartana, és összefogná a jelenleg teljesen szervezetlen klubokat. Azt is javasolták, hogy mi legyünk a kezdeményezők.* Ennek okául azt mondták, hogy egyrészt mi példát mutatunk a jó helyi szervezésből, másrészt, hogy ha valamelyik nagy amerikai klub lenne a szervező, akkor ezt a többi nagy klub nem fogadná örömmel.

Azt hiszem, hogy ez nagyon jó gondolat! *Kérem ezért mindazoknak a kluboknak a vezetőit, amelyek ezzel egyetértenek, keressenek meg, és kezdjük el!*

DR. SIMONYI ENDRE

Sztringkereső

Sok esetben szükség lehet arra, hogy a memóriában szövegeket gyorsan megkeressünk. A feladat BASIC-nyelven is megoldható, de nagyon lassan. Az assembler program ezt a lassúságot szünteti meg.

A program a Homelab—3-mal használatos EDITOR ASSEMBLER segítségével készült, amely a gép alapszintű tudásához tartozik. A program három részből áll. Az első rész a keresett szöveget kéri be, amit a 4400H címre tárol. A keresett adat minimális hossza két betű. A program második része a tulajdonképpeni kereső rutin. Az itt megadott adatok 28 kbájt vizsgálatát teszik lehetővé, de természetesen a számláló átírásával nagyobb területet is vizsgálhatunk.

A rutin 40EDH-tól 5 bájtot rendszerváltozó részére tart fenn. A program harmadik része egy értékelő rutin, amely kiírja a megtalált sztring címét hexa kódban, és csipogással jelzi a keresés eredményét.

A program négy belső rutint használ fel. Az egyik a sztringet gyűjti be a billentyűzetről (CALL 0546H), a másik táblázatból szöveget ír ki (CALL 0188H), a harmadik egy hangot képez (CALL 18 E1H), a negyedik pedig kiírja a DE regiszter tartalmát hexadecimális formában.

A program továbbfejlesztése már a memóriában található adatok formátumától függ.

BALOGH TIBOR

```
                1               ;STRING KERESO
2                               ;KESZITETTE BALOGH TIBOR 1986.7.27
3                               ORG  40EDH
4                               LOAD 40EDH
5                       KC:     EQU  4500H  ;A VIZSGALT TERULET ELEJE
6                       VC:     EQU  6FFFH  ;A VIZSGALT TERULET HOSSZA
7                       RVALT:  DS   5
8  40F2 FF4D4547        AKAD:   DB   0FFH,'MEDVAN',0A0H
9  40F6 56414EA0
9  40FA 4E494E43        NKAD:   DB   'NINCS',0A0H
9  40FE 53A0
10                      ;
11                      ;A KERESETT STRING BEKERESE
12                      ;
13 4100 3E0C                    LD   A,0CH
14 4102 EF                      RST  40     ;KEPERNYO TORLES
15 4103 CD4605                  CALL 0546H  ;SORBEUEVO RUTIN
16 4106 21FF43                  LD   HL,43FFH;PUFFER ELOTTI CIM
17 4109 0EFE                    LD   B,0FEH ; SZAMLALO
18 410B 77              ISM:    LD   (HL),A
19 410C 04                      INC  B
20 410D D7                      RST  16
21 410E 23                      INC  HL
22 410F A7                      AND  A
23 4110 20F9                    JR   NZ,ISM
24                      ;
25                      ;KERESES RUTIN
26                      ;
27 4112 210044                  LD   HL,4400H;PUFFER ELEJE
28 4115 22ED40                  LD   (RVALT),HL
29 4118 21EF40                  LD   HL,RVALT+2
30 411B 70                      LD   (HL),B
31 411C 210045                  LD   HL,KC
32 411F 01FF6F                  LD   BC,VC
33 4122 ED50ED40        CPI:    LD   DE,(RVALT);KERESETT SZOV. CIME
34 4126 1A                      LD   A,(DE)
35 4127 EDB1                    CPIR
36 4129 E24C41                  JP   PO,NEM ;NEM TALAL
37 412C A7                      AND  A
38 412D E5                      PUSH HL
39 412E C5                      PUSH BC
40 412F ED4BEE40                LD   BC,(RVALT+1);STRING HOSSZA MI
41 4133 05              CIKL:   DEC  B
42 4134 13                      INC  DE
43 4135 1A                      LD   A,(DE)
44 4136 EDA1                    CPI         ;UJABB BETU
45 4138 201C                    JR   NZ,UJRA;A KOV. BETU NEM EGYEZIK
46 413A 78                      LD   A,B
47 413B C5                      PUSH BC
48 413C 0600                    LD   B,00H
49 413E B8                      CP   B
50 413F C1                      POP  BC
51 4140 2803                    JR   Z,CR2
52 4142 C33341                  JP   CIKL
53 4145 C1              CR2:    POP  BC
54 4146 E1                      POP  HL
55 4147 3EFF                    LD   A,255  ;MEGVAN
56 4149 C34E41                  JP   VEGE
57 414C 3E00            NEM:    LD   A,128  ;NINCS
58 414E 2B              VEGE:   DEC  HL
59 414F 22F040                  LD   (RVALT+3),HL;VEGCIM
60 4152 CD5B41                  CALL KIIRO
61 4155 C9                      RET
62 4156 C1              UJRA:   POP  BC
63 4157 E1                      POP  HL
```

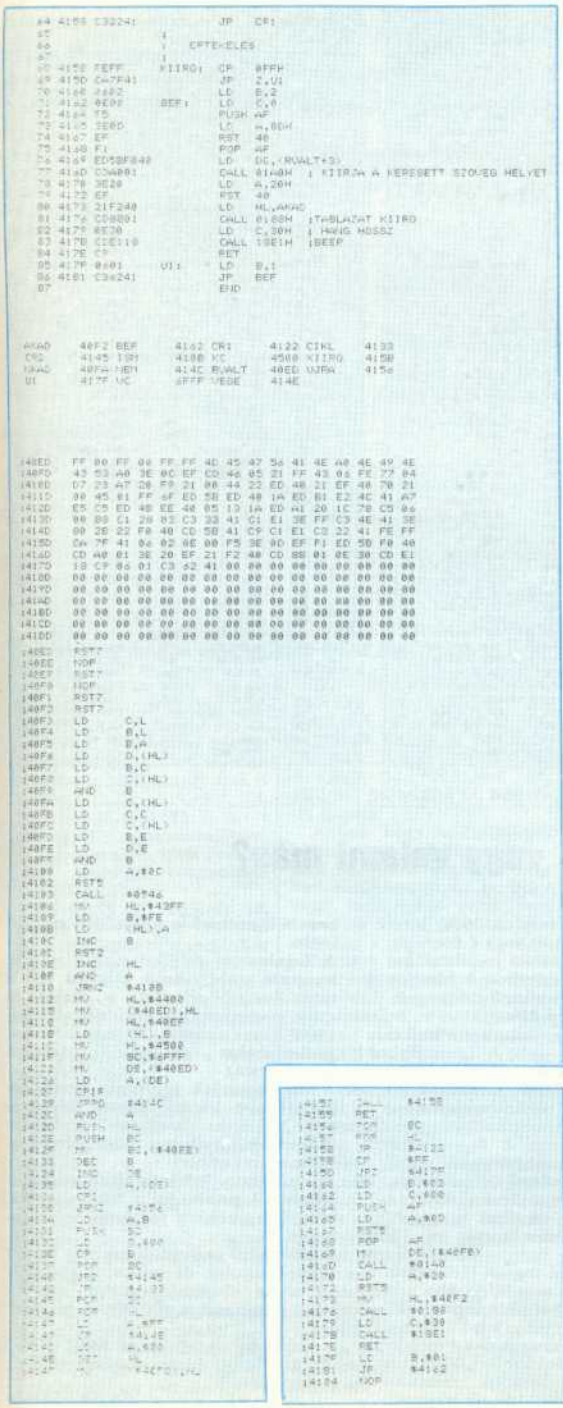
COMMODORE 64

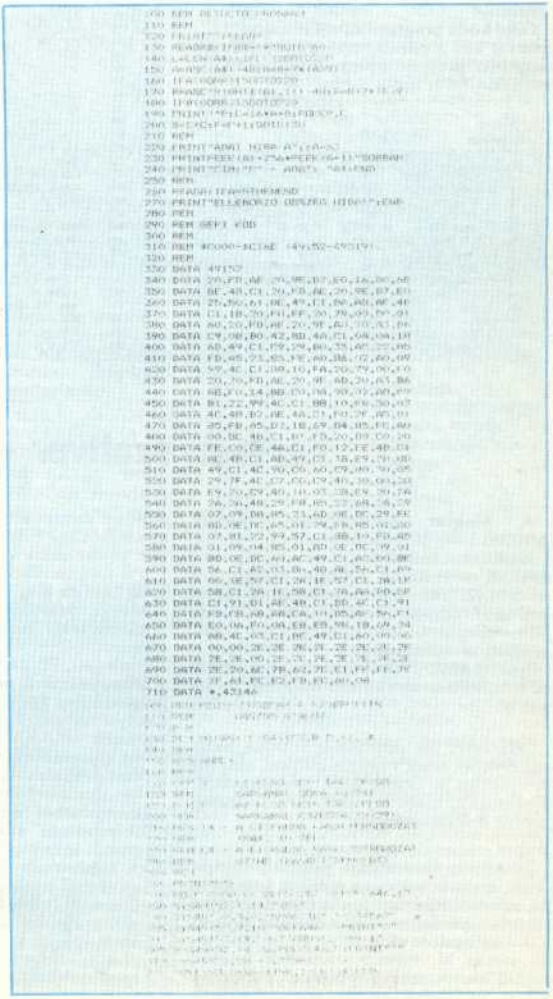
Eredeti karakterek négyszeres nagyításban

Munkám során többször előfordult, hogy nagyobb méretű karaktereket kellett volna megjeleníteni a képernyőn, komolyabb szoftvertámogatás nélkül. Erre a célra fejlesztettem ki egy gépi kódú szubrutint, amely az eredeti karaktereket négyszeres nagyításban jeleníti meg.

Minden jel 4*4, azaz 16 karakter helyet foglal el a képernyőn, így négy sorban maximum 10 karakter fér el. A megjelenítendő karakterek színe egyenként beállítható. A betöltőprogram lefuttatása után a szubrutin SYS utasítással hívható. A paraméterek megadási formája az alábbi mintaprogramban megtalálható.

SPÁNYIK JÁNOS





PRIMO

Kör és egyenes rajzolása

BASIC-ből a következő módon lehet meghívni a rajzolórutinokat.

Kör rajzolása esetén *(1. program)*

POKE CIM+247,X,Y,R

A= CALL (CIM)

ahol CIM a rutin kezdőcíme, X és Y a kör középpontja, R a rádiusz.

Egyenes rajzolásakor *(2. program)*

POKE CIM+137,X1,Y1,X2,Y2

A= CALL (CIM)

ahol CIM a rutin kezdőcíme, X1 és Y1 az egyenes kezdőpontja, X2 és Y2 pedig a végpontja.

Gépi kódú programból a TB, illetve TBL címekre kell a paramétereket megadni. Az assembler program operandusai hexadecimálisak.

HOSSZÚ LAJOS

1. program

```
    1    LD HL,TB+2   78      LD B,H
    2    XOR A        79      LD L,(IY-4)
    3    CP (HL)      80      LD H,A
    4    RET Z        81      ADD HL,HL
    5    INC HL       82      POP DE
    6    LD (HL),A    83      ADD HL,DE
    7    INC HL       84      DEC HL
    8    LD (HL),A    85      LD (IY+2),L
    9    INC HL       86      LD (IY+3),H
   10    LD (HL),A    87      ADD HL,BC
   11    LD IY,TB+6   88      XOR A
   12 K1:LD A,(IY-6)  89      SBC HL,DE
   13    LD B,(IY-5)  90      LD (IY+4),L
   14    LD H,(IY-4)  91      LD (IY+5),H
   15    LD L,(IY-3)  92      BIT 7,B
   16    PUSH AF      93      JR Z,K6
   17    ADD A,H      94      XOR A
   18    JR C,K2      95      LD L,A
   19    LD E,A       96      LD H,A
   20    LD A,B       97      SBC HL,BC
   21    ADD A,L      98      LD C,L
   22    LD D,A       99      LD B,H
   23    CALL NC,83   100  K6:LD E,1
   24    LD A,B       101     LD D,A
   25    SUB L        102  K7:PUSH DE
   26    LD D,A       103     SLA E
   27    CALL NC,83   104     LD D,B
   28 K2:POP AF       105     LD HL,TB+6
   29    PUSH AF      106     ADD HL,DE
   30    SUB H        107     LD E,(HL)
   31    JR C,K3      108     INC HL
   32    LD E,A       109     LD D,(HL)
   33    LD A,B       110     BIT 7,D
   34    ADD A,L      111     JR Z,K8
   35    LD D,A       112     XOR A
   36    CALL NC,83   113     LD L,A
   37    LD A,B       114     LD H,A
   38    SUB L        115     SBC HL,DE
   39    LD D,A       116     EX DE,HL
   40    CALL NC,83   117  K8:EX DE,HL
   41 K3:POP AF       118     POP DE
   42    PUSH AF      119     XOR A
   43    ADD A,L      120     SBC HL,BC
   44    JR C,K4      121     JR NC,K9
   45    LD E,A       122     LD D,E
   46    LD A,B       123     ADD HL,BC
   47    ADD A,H      124     LD C,L
   48    LD D,A       125     LD B,H
   49    CALL NC,83   126  K9:LD A,E
   50    LD A,B       127     CP 2
   51    SUB H        128     JR Z,K10
   52    LD D,A       129     INC E
   53    CALL NC,83   130     JR K7
   54 K4:POP AF       131 K10:LD A,D
   55    SUB L        132     LD E,D
   56    JR C,K5      133     SLA E
   57    LD E,A       134     LD D,B
   58    LD A,B       135     LD HL,TB+6
   59    ADD A,H      136     ADD HL,DE
   60    LD D,A       137     LD E,(HL)
   61    CALL NC,83   138     INC HL
   62    SUB H        139     LD D,(HL)
   63    SUB H        140     LD (IY-2),E
   64    LD D,A       141     LD (IY-1),D
   65    CALL NC,83   142     CP 0
   66 K5:XOR A        143     JR Z,K12
   67    LD H,A       144     CP 1
   68    ADD HL,HL    145     JR Z,K11
   69    INC HL       146     INC (IY-3)
   70    EX DE,HL     147 K11:DEC (IY-4)
   71    LD L,(IY-2)  148     JR K13
   72    LD H,(IY-1)  149 K12:INC (IY-2)
   73    PUSH HL      150 K13:LD A,(IY-4)
   74    SBC HL,DE    151     CP (IY-3)
   75    LD (IY+0),L  152     JP NC,K1
   76    LD (IY+1),H  153     RET
   77    LD C,L       154  TB:DS 6C
```

2. program

```
    1    LD HL,TBL    53      LD E,B
    2    LD E,(HL)    54      SRL E
    3    INC HL       55      LD D,0
    4    LD D,(HL)    56      PUSH DE
    5    INC HL       57      EXX
    6    LD B,(HL)    58  U8:EXX
    7    INC HL       59      LD H,B
    8    LD C,(HL)    60      LD L,C
    9    LD A,B       61      POP DE
   10    CP E         62      PUSH DE
   11    JR NZ,U1     63      ADD HL,DE
   12    LD A,C       64      LD D,0
   13    CP D         65      LD E,B
   14    JP Z,83      66      SBC HL,DE
   15 U1:LD L,1       67      JR C,U10
   16    LD A,B       68      EX (SP),HL
   17    SUB E        69      EXX
   18    JR NC,U2     70      XOR A
   19    LD A,E       71      CP C
   20    SUB B        72      JR NZ,U8
   21    LD L,0FF     73      LD A,D
   22 U2:LD B,A       74      ADD A,H
   23    JR NZ,U3     75      LD D,A
   24    DEC L        76      JR U11
   25 U3:LD H,1       77  U8:LD A,E
   26    LD A,C       78      ADD A,L
   27    SUB D        79      LD E,A
   28    JR NC,U4     80      JR U11
   29    LD A,D       81 U10:ADD HL,DE
   30    SUB C        82      EX (SP),HL
   31    LD H,0FF     83      EXX
   32 U4:LD C,A       84 U11:XOR A
   33    JR NZ,U5     85      CP C
   34    DEC H        86      JR NZ,U12
   35 U5:PUSH BC      87      LD A,E
   36    CALL 83      88      ADD A,L
   37    POP BC       89      LD E,A
   38    LD A,0BF     90      JR U13
   39    SUB D        91 U12:LD A,D
   40    LD D,A       92      ADD A,H
   41    LD A,B       93      LD D,A
   42    CP C         94 U13:PUSH BC
   43    JR C,U6      95      CALL 83
   44    PUSH BC      96      POP BC
   45    LD C,0       97      LD A,0BF
   46    JR U7        98      SUB D
   47 U6:LD B,C       99      LD D,A
   48    LD C,A      100      DJNZ U8
   49    PUSH BC     101      POP DE
   50    LD C,1      102      RET
   51 U7:EXX         103 TBL:DS 4
   52    POP BC
```

Hardver, szoftver — vagy valami más?

A Magyar Szabványügyi Hivatal (MSZH) 1979 óta rendszeresen foglalkozik a számítástechnikai fogalmak egységesítésével. A nemzetközi előírások alapján készülő MSZ 7788 „Az adatfeldolgozás fogalmai" szabványsorozat eddig közel 1500 fogalom elnevezését és meghatározását (definícióját) szabványosította. Az 1979-ben kidolgozott MSZ 7788/1 szabvány tartalmazza a számítástechnika két alapvető fogalmát is:

„Hardver: Az adatfeldolgozásban használt berendezések vagy azok részei, megkülönböztetve a számítógépi programoktól, eljárásoktól, szabványoktól és a hozzájuk tartozó dokumentációtól."

„Szoftver: Számítógépi programok, eljárások, szabványok és az ezekhez kapcsolódó, az adatfeldolgozó rendszer működésére vonatkozó dokumentációk összessége."

A két fogalom elnevezése széles körű vitát váltott ki, többen javasolták magyar nyelvű elnevezések bevezetését. Az eddig ismert javaslatok egyike sem tekinthető minden szempontból megfelelőnek, ezért az olvasókhoz fordulunk, hogy e két fogalom elnevezésére vonatkozó **javaslataikkal** a lap megjelenésétől számított **1 hónapon belül** küldjék meg a következő címre:

Magyar Szabványügyi Hivatal

villamossági osztály, számítástechnikai csoport

Budapest, Pf. 24. 1450

Útmutatásul a megfelelő javaslatok kidolgozásához közlünk néhány szempontot, amelyet célszerű figyelembe venni az elnevezésnél:

1. Az elnevezés (fogalomnév) legyen alkalmas a fogalom **egyértelmű azonosítására.** (A „hardver" helyettesítésére ebből a szempontból nem alkalmas a „gép", „berendezés" stb., mert számos más fogalmat is felölel.)
2. A fogalomnév **rövid** legyen (két-három szótagnál lehetőleg nem hosszabb).
3. **Összetett** (több szóból álló) **fogalomnévnél** kerülni kell a birtokos jelző alkalmazását (pl. „a számítástechnika berendezése"), helyette inkább mellékneves szerkezetet alkalmazzunk (pl. „számítástechnikai berendezés"), amely könnyebben beilleszthető a különböző mondatokba.
4. A fogalomnév feleljen meg a magyar **hangtani szabályoknak** (kerülni kell a mássalhangzó-torlódást, a magas és mély magánhangzók egyidejű alkalmazását, a csak azonos, illetve hasonló magánhangzókat — pl. *e* és *é* — tartalmazó szavakat stb).
5. A fogalomnevek legyenek alkalmasak **foglalkozást jelentő főnevek képzésére** (pl. hardveres: hardverrel foglalkozó szakember; szoftveres: szoftverrel foglalkozó szakember).
6. A fogalomnevek legyen alkalmasak **melléknevek képzésére** (pl. a hardvertermék, szoftverjegyzék stb. fogalmak helyettesítésére).

A különböző szempontokat még tovább is lehetne sorolni, de sajnos, aligha lehet majd olyan elnevezést találni, amelyek minden szempontból egyformán megfelelnének.

A beérkező javaslatok az MSZ 7788/1 soron következő korszerűsítésekor kerülnek megvitatásra.

GYŐRI JÁNOS

BITEK ÉS FIGURÁK

Állásértékelés VII.

KIRÁLYTÁMADÁS ÉS KIRÁLYBIZTONSÁG

A következő játszma szép példa arra, hogyan támad világos a meggyengült királyállásra, hogyan fosztja meg világos és sötét királyállást a gyalogjaitól, és végül hogyan viszi döntőre támadását a védtelen sötét királlyal szemben. (Balla Z.—R. Spielmann, Budapest, 1928.)

1. Hf3,Hf6 2. c4,c6 3. Hc3,d5 4. d4,dc: 5. a4,e6 (Abban az időben e változatot még nem ismerték. Ma már mindenki tudja, hogy 5.—,Ff5 a legjobb.)

6. e3 (még erőteljesebbnek látszik a 6.e4)

6. —,Fb4 7. Fc4:,0—0 8. 0—0,b6 9,Ve2,Fb7 10. Bdl,Hbd7 11. e4!,Vc8 12. e5,Hd5 13. He4,a5 (Világos centrális előrenyomulása a d6 pont megszerzésére irányul, de a Hf6 előzése után királyszárnyi támadás is lehetséges. Mindez a d4 és d5 pontok meggyengülése árán, amit azonban egyelőre nyugodtan el lehet viselni. Sötét utolsó lépése előkészíti a futó visszavonulását anélkül, hogy megengedné a d4—d5-öt.)

14. Vc2!,Fe7 15. Fg5!,f6 (Kényszer, mert a futócsere után a d6-ra kerülő huszár idegen test lenne sötét állásában.)

16. ef:,gf: (Másképp az e5 pont válik támaszponttá a világos tisztek számára. A centrumharc következménye a sötét királyállás meggyengülése.)

17. Fh6,Bf7 18. Bac1,Hf8 19. Fb3,Vd7 20. Hg3,Hg6 21. Be1!,Ff8 22. Fd2,Be8.

23. Be4! (Fenyeget a bástyakettőzéssel és Hf5-tel. Egyben a bástya a királyszárny felé is kacsint.) **23. —,c5 24. Bce1,Hc7 25. Bg4,Ff3:** (Sötét is ront egyet világos sáncállásán, és gyalogot nyer. Sötét gyengeségei azonban könnyebben hozzáférhetők.)

26. gf:,f5 27. Bg5,cd: 28. Vd3,Fg7 29. Hh5,Kh8 (Fh6?-ra most, vagy előbb: Bg6:†!. Két tisztet nyer a bástya ellen.)

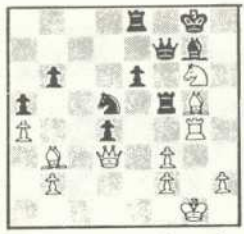
30. Bg3,f4? (Sötét helyzete nem könnyű. A tett lépéssel világos vezérfutóját akarja kizárni, de világos szellemes minőségáldozattal fokozza a támadás erejét.)

31. Bg6:,hg: 32. Hf4:,g5 (Jobb volt azonnal visszaáldozni a minőséget 32.—, Bf4:-gyel és a kiegyenlítésért küzdeni.)

33. Hg6†,Kg8 34. Fg5:,Hd5 35. Be4!,Bf5 36. Bg4! (A bástya ugyanazon az úton, mint kollégája megérkezik a g-vonalra, a döntő események színterére.)

36. —,Vf7 *(1. ábra!!)*

27. Fh6!!,Bf3? (Jobb lett volna az azonnali vezéráldozat, 37. —,Fh6: 38. He5†,Vg7 stb. A tett lépéssel világos hamarosan tisztelőnyre tesz szert.)



a b c d e f g h

A sötét király g8-on áll.

38. Vd1!,Bf2: (Most már nincs más!)

39. He5!,Bf1† (Kényszeráldozat.)

40. Vf1:,Vf1:† 41.Kf1:Kh7! 42. Bg7:† (Fg7:† nem jó He3† miatt.)

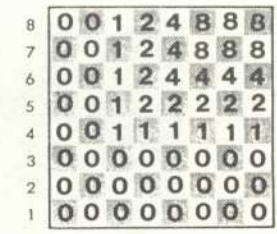
42. —,Kh6 43. Bg6†,Kh7 44. Fd5:,ed: 45. Bg5,Bc8 46. Hd3!,Bc4 47. Bd5:Ba4: 48. h4,Bc4 49. Ke2,a4 50. Bb5,Bc2† 51. Kf3,a3! 52. ba:,Bc3 53. Ke4,Ba3 54. Bb6:,Ba1 55. Hf4,Bh1 56. h5,Bh4 57. Bd6 sötét feladta.

Ahhoz, hogy az ellenséges király ellen eredményes támadást indítsunk, pontosan meg kell határoznunk azt is, hogy saját királyunkat ezzel nem tesszük-e ki nagyobb veszélynek. Ezt a sakkozók a konkrét változatok számításán kívül nagy tapasztalatuk miatt intuitív módon megérzik. A program számára ez nem járható út, mert csak konkrét mennyiségekkel tud számolni. A program kiszámítja az előbb említett módon a királytámadás mértékét, és valamilyen módon megállapítja a királybiztonság értékét. Ennek a két számnak az aránya mondja meg a programnak, hogy támadjon-e vagy védekezzen.

A királybiztonsági érték kiszámításánál a következő szempontokat kell figyelembe venni:

1. A királyt saját gyalogjainak és figuráinak a legerősebben kell védeniük.

2. A gyalogok biztonságosabb védelmet nyújtanak, mint a tisztek.



3. A védelmet minél jobban támogatjuk, annál biztonságosabb.

4. A védő figura minél közelebb van királyához, annál hatékonyabb a védelme.

Az előbbi szempontokat könnyen meghatározhatjuk a számítógép számára érthetően, kvantitatív módon.

A királyt védő minden figurára a királyvédelem értékét a következő képlettel érdemes meghatározni:

KIRÁLYVÉDELEM = SAJÁT FIGURA KÖZELSÉG / MÓDOSÍTOTT FIGURAÉRTÉK

A SAJÁT FIGURA KÖZELSÉG értékét a *2. ábra* alapján számítjuk: 8 a saját király melletti mezőkre, 4 az ezeket körülvevőkre, 2 a következő gyűrűre, 1 az utolsóra. Az ezeken a mezőkön túl lévők már túl távoliak a védekezés szempontjából, és ezért zérus a súlyozóértékük.

MÓDOSÍTOTT FIGURAÉRTÉK =
- = figura értéke, ha a figura nincs megvédve;
- = figura értéke. (n—1), ha a figurát n db saját figura védi, beleértve a gyalogosokat és a királyt is.

Ezt a képletet használva kiszámíthatjuk minden figurára a királyvédelem értékét, és ezeket összeadva megkapjuk a királybiztonsági mutatót. Ezt mind a világos, mind a sötét királyra kiszámítva, a kettő arányából kikövetkeztethetjük, hogy melyik fél királyállása a gyengébb.

Ezeket az értékeket az előbb bemutatott játszmákra kiszámítva láthatjuk, hogy a bemutatott képlet híven tükrözi a valóságos királybiztonságot. Ezen felül ezt a képletet is lehet csiszolni, finomítani. Javíthatunk még az értékelésen, ha külön figyelembe vesszük az alábbiakat.

1. A király előtti gyalogállást és a következő táblázat szerinti pontokat adjuk hozzá az előbbiek alapján kiszámított királybiztonsági mutatóhoz.

Ha a gyalog a király melletti oszlopban helyezkedik el:

Gyalog elhelyezkedése a király előtt	Pontérték
1. sorban	+2
2. sorban	−2
3. sorban	−4

Ha a gyalog a királlyal egy oszlopban helyezkedik el:

Gyalog elhelyezkedése a király előtt	Pontérték
1. sorban	+4
2. sorban	−4
3. sorban	−6

2. Ha a királyunk vonalában nincsen ellenséges gyalog, akkor a kinyílt vonal miatt királyállásunk romlott, és ezért büntetőpontot kell levonni. Ennek értéke az előző pontokhoz viszonyítva a −4 és −8 közé kell hogy essen.

3. A királybiztonsági mutatónál is figyelembe vehetjük a sáncolási jog elvesztését, a következő pontozással:
- — ha elsáncolt: +6
- — sáncolási jogát elvesztette: −12.

4. A középjátékban a királyra ne számoljuk ki a centrumértékeket, mert ez csak lassítja a számítást, és a király lépéseit is felpontozza annak ellenére, hogy azok csak megtarják az állást, de nem javítják. Vagyis nagyobb az esély arra, hogy a program a kicsit bizonytalan de aktív lépés helyett a biztos királylépést választja.

5. Bizonyos lépésszám után vagy gyenge ellenséges haderőtámadás esetén, az alapsori matt elkerülése érdekében egy kis többletet adhatunk a pontszámhoz, ha az utolsó soron fenyegető matt elkerülése érdekében a királynak rendelkezésére áll egy kibúvó mező. Ez világos esetében rendszerint a h2 vagy g2, sötét esetében a h7 vagy g7.

KOVÁCS P. ATTILA

GÉPI ELLENFELEINK

Mephistók a világ élvonalában

Cikksorozatunkban folyamatosan ismertetjük azokat a sakkszámítógépeket, amelyek elsősorban a nyugati országok üzleteiben, áruházaiban kaphatók. Figyelemmel kísérjük az újonnan megjelenő típusokat, de szólunk azokról a régebbi készülékekről is, amelyek még nem avultak el, nem tűntek el a piacról.

Új üzleti stratégia

A legutóbbi egy-két esztendő tapasztalatai alapján kijelenthetjük, hogy a sakkszámítógépek öt-hat gyártójának versenyében a müncheni Hegener+Glaser cég tört az élre, s napjainkban a Mephisto készülékek — elsősorban Európában — vitathatatlanul a legnépszerűbbek. Ezt persze nem úgy kell érteni, hogy minden Mephisto típust minden szempontból — játéktudást, kivitelt, árat figyelembe véve — például az USA-beli Fidelity, vagy a hongkongi NOVAG készülékei fölé helyezünk. De úgy igen, hogy Hegener+Glasernek a legnagyobb a választéka, s nincs az a kategória, amelyben a sokféle igénnyel fellépő vásárló ne tudná megtalálni a maga számára legmegfelelőbb készüléket.

Eléggé egyedülálló, hogy a cég régebbi típusai, a Mephisto II és III még mindig piacképesek — az előbbinek „kortársai", például Fidelity akkori Chess Challengerei szinte mind elavultnak tekinthetők —, s ezért szükségesnek is tartottuk ismertetésüket, pedig az azóta eltelt mintegy három éven át Hegenerék tucatnál is több újabb típust hoztak forgalomba. A cég sikerét két tényezőnek köszönheti, amelyekkel vetélytársai sorából kiválik: a már említett moduláris rendszernek, s egy igen bátor és merőben új üzleti stratégiának. Ez abban áll, hogy 1983-ban — talán mert nem voltak mindenben elégedettek a Nitsche—Henne programozópár eredményeivel — megnyitották kapuikat a szó szoros értelmében minden programozó előtt. (A vezető világcégek gyakorlatában ez újítás volt, és ma is szinte egyedülálló; amint látni fogjuk, a többi „nagy", az említett Fidelity és NOVAG vagy SciSys mind egyetlen programozó, illetve programozócsoport függvénye.) Senkivel sem kötöttek kizárólagossági megállapodást. Ez a sakkprogramozók számára is tulajdonképpen igen kedvező, nem nehéz a céghez „bejutni", csak épp élenjáró programmal kell hozzá. Ez az információnk a vásárló számára is sokatmondó, mert Hegenerék gyártási stratégiája fokozza készülékeik sokféleségét, színesíti választékukat. Az egyes programok legfőbb sajátosságaira a következőkben kitérünk.

Richard Lang és Ossi Weiner a Mephisto Cologne-nyal

Modular, Exclusive és München

Az igények növekedése szükségessé tette, hogy új típusú hardvereket is tervezzenek. Széles körben elterjedtek a „sakktábla-számítógépek", vagyis azok a készülékek, amelyek sakk-készlet formáját öltik, és a lépés megtételével a táblán a számítógép közvetlenül „beveszi" azt. A sakktáblán lévő diódák kigyulladása jelzi az ellenfél válaszlépéseit is. Vitathatatlanul ez a legkényelmesebb kezelési mód, s azokat a készülékeket, amelyekhez külön sakk-készletet kellett használni, nem fejlesztették tovább. Megszakadt a korábbi Mephisto-sorozat is, noha az ESB tábla jelezte a jövő fejlődés útját, de túl drága volt, és Mephisto IV program már nem született. Ehelyett 1983 végére kialakítottak két új sakk-készleti hardvert, a Mephisto Modulart és Mephisto Exclusive-ot. A kettő szerkezetileg azonos, mágneses szenzorokkal működik, természetesen ismét moduláris rendszerű, különbség csak kivitelükben van. A Modular 30×35×3,5 cm, az Exclusive 41×41×4 cm méretű táblából, hozzájuk tartozó bábkészletből, a fiókjukban elhelyezett — a régi Mephistóhoz hasonlatos —, nyomógombokkal ellátott programkazettából és négy betűs kijelzőből áll. A Modular-tábla és -figurák műanyagból, az Exclusive-nál ugyanezek szépen faragott fából készültek. A programkazetta cserélhető!

Harmadikként csatlakozott a két hardverhez 1985-ben a München. A cég ezzel újabb elit készüléket jelentetett meg, az ESB-hez hasonlót. Mérete csaknem pontosan ugyanakkora: a tábláé 50×50×6 cm, 2 cm-rel alacsonyabb az ESB-nél. Ez érthető, hiszen ennek fiókjában a keskenyebb programkazetta foglal helyet, az ESB-be a régi „fekete dobozt" kell beletenni. Bábjai is pontosan ugyanakkorák; a legnagyobb figura, a király 9 cm magas. Összehasonlításul: az Exclusive királyának mérete 7 cm.

Még a München megjelenése előtt — kísérleti jelleggel az 1983. évi budapesti vb-n, sériagyártásban a következő esztendőben — megjelent a Modularnak és az Exclusive-nak „S" jellel ellátott verziója, majd természetesen a Münchené is. Ezek igen lényeges újítást tartalmaznak: a korábbi 6502-es vagy 1806-os típusú, 8 bites processzor helyett 68000-es, 16 bites processzor működteti őket. Ennek programozástechnikai jelentőségére talán szükségtelen rámutatnunk; a hardver az új processzorral jelentősen nagyobb működési sebességet is biztosított.

A programok

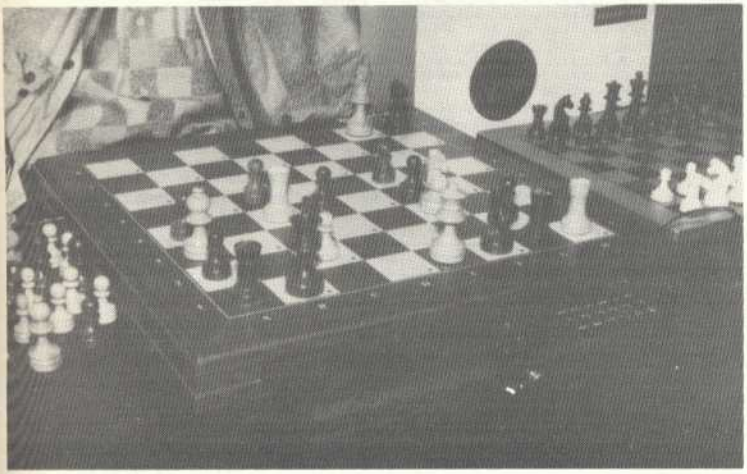
Nem térünk el nagyon a valóságtól, ha azt mondjuk, hogy mindhárom készülékbe bele lehet helyezni mindazokat a legsikerültebb programokat, amelyek a Mephisto III óta készültek. Szeretnénk azonban ennek lehetőségeiről az olvasót kissé ponto-

sabban tájékoztatni, nehogy azt higgyék: a Modular, Exclusive, illetve München elnevezés — „S" jellel vagy anélkül — fedi a készülék típusát. Egyébként a cég is, amikor egy-egy programmódosítással vagy új programmal kijött, igyekezett ennek mindig újabb megkülönböztető jelzést adni. Ezek szövevényében azonban nem könnyű kiismerni magunkat.

A Modular és az Exclusive eredetileg a Mephisto III programot tartalmazták, amely új köntöseiben némileg eredményesebben működhetett, mert a korábbi 6,1 helyett 8 MHz-en futott. Az „S" kivitelűek ugyancsak, de a programot Nitsche és Henne a nagy teljesítményű processzor lehetőségeit kihasználva, jelentősen továbbfejlesztették. Az Exclusive „S" a javított III programmal kiválóan szerepelt 1984-ben a glasgow-i vb-n: holtversenyben az elsők között végzett, és elnyerte a kereskedelmi forgalomban lévő legjobb készülék díját. A számítógép sikeresen mutatkozott be nálunk is: az egész világ szaksajtóját bejárt játszmában győzött egy, a KSH-ban rendezett mérkőzésen Bíró András fiatal mesterjelölt ellen.

1985-ben a Mephistók óriási lendülettel, új és továbbfejlesztett programok egész sorával törtek az élre. Elsőként az ún. B+P (Blitz+Problem) modul jelent meg, amely mindhárom alapkészülékbe behelyezhető; ezzel jelent meg legelőször a München is. A program alkotója a svéd Ulf Rathsman, aki korábban a „Conchess" típusú számítógépek programjait készítette, de ez a cég tönkrement. Utána Princhess, majd Plymate néven jelentkezett kitűnő programjaival, amelyek végül Hegeneréknél leltek jó gazdára. Sokáig azonban ez a B+P modul nem maradt a piacon, mert felváltotta a Mephisto Modular II, amely a B+P-nek továbbfejlesztett változata, és azóta is a gyár egyik kiváló, standard terméke.

A nagy sláger azonban csak ezután következett. Az angol Richard Lang, aki már korábban sikeresen kísérletezett a 68000-es processzor alkalmazásával ("PSION" programjai személyi számítógépekre kerültek forgalomba), megbízást kapott a Hegener+Glasertől, hogy készítsen új programot az Exclusive S és München S hardverekre. 12 MHz sebességűre gyorsította őket, és minden idők egyik legnagyobb számítógépes sakksikerét könyvelte el: az új programjával ellátott három készülék az első három helyet foglalta el a szeptemberi amszterdami mikro vb-n! Kétségtelen, hogy a Mephisto Amsterdam bármely alapkészülékben jelenleg a világ legerősebben sakkozó mikroszámítógépe. Az 1983—1985 közötti időszakban további Mephisto készülékek is megjelentek.



Mephisto Exclusive és Modular

Más típusok, árak

Elsőként a Mephisto Mondialt említjük, amelynek programját a holland Frans Morsch készítette. Az amszterdami vb amatőrcsoportjában volt győztes a Nona, amelyet Manfred Hegener cégfőnök azon nyomban megvásárolt cége számára. A Mephisto Mirage érdekes módon a régi Mephisto II ESB program átalakított formáját rejti: processzora a korábbi 6,1 helyett szintén 8 MHz-en fut. A program a Mobil nevő zsebsakkba is beilleszthető. Nem hagyjuk ki a sorból a Teufelchent sem, amely a legolcsóbb Mephisto, kivitele igénytelen és tudása sem mondható kiemelkedőnek; David Levynek egy régebbi programját rejti.

Amikor e sorokat írjuk, előkészületben van az MM3. Programozója a holland Ed Schröder, aki Rebel programjával a közelmúltban zajlott kölni vb-n ért el kiváló eredményt. Ugyanannyi pontot szerzett, mint három másik kísérleti Mephisto-program: az amszterdaminak „Cologne" változata; a legújabb Plymate és Nona továbbfejlesztése, amelyet a Supermundialban látunk viszont.

Minden jó sakkozónak is a legmelegebben ajánlhatjuk az MM2 típust, vagy akár a Modulart a III. programmal: áruk 760—800 nyugatnémet márka. Ugyanezek a programok az Exclusive-ben 100, a Münchenben 1500 márkába kerülnek. Az MM2 modul önmagában 350—400 márka; csere esetén kevesebb. A Mondial és a Mobil ára 350, a Mirage-é 500, a Teufelchené 150 márka. A világbajnokért borsos árat kell fizetni: az Exclusive-ben 3000, a Münchenben 3500 márkát, aminek az a magyarázata, hogy csak maga a modul — magas technikai paraméterei miatt — 2400 márkáért cserélheti gazdáját, sakkrajongóra a kereskedőt.

LINDNER LÁSZLÓ

Felajánlunk megvételre használt, üzemképes állapotban lévő számítástechnikai berendezéseket:

19 db DZM 180 típusú nyomtató
10 db VT 61400 GDN (modem)
7 db TAP—34 berendezés (képernyős display, floppyegység)
1 db Soemtron 415 típusú feliratos lyukkártyagép

DATORG

Érdeklődni lehet a Bp. V., Dorottya u. 6. szám alatti irodaházban, Ferenczi Sándor munkatársunknál.
Telefon: 177-403

COMMODORE 64

Ha nehéz a Quasimodo (avagy Hanch Back)

Úgy hírlik, hogy Magyarország és Jugoszlávia élen jár a programfejtésben. Nem nehéz megérteni, miért. A játékprogramok hasábjain ezentúl szívesen közlünk az alábbihoz hasonló programokat, amelyek megkönnyítik az olykor soha véget nem érő játékok végigjátszását.

A program a feliratok megjelentetése után betölti a Quasit, majd elvégzi a változtatásokat, és az I gomb megnyomása után elindítja a játékot. Ha valakinek kazettán van meg a programja, vagy más néven szerepel, az a 23-as sorban elvégezheti az átalakítást.

A működés elve. Egy gépi kódú programmal megkerestük a játék azon helyeit, ahol a sprite-sprite ütközést ellenőrzi, majd itt megfelelően átírjuk az utasítást. A kereszteseket nem sprite-okkal oldja meg, ezért őket továbbra is át kell ugrani (azért egy kis izgalom nem árt)!

ILLÉS PÉTER

```
1 REM ********************************
2 REM *   QUASIMODO EASIER VERSION    *
3 REM ********************************
4 REM *(C) PIPROG SYSTEM HOUSE , 1984*
5 REM ********************************
6 REM * WRITTEN BY                    *
7 REM *                               *
8 REM * FURSTNER JÁNOS & ILLÉS PÉTER *
9 REM ********************************
10 ON A GOTO 24
11 POKE 53280,9:POKE 53281,7:PRINT"■■";
12 PRINT"■■  QUASIMODO (ALIAS HUNCH BACK) LOADER   ";
13 PRINT"■                                          ";
14 PRINT"■WRITTEN BY  FURSTNER JÁNOS & ILLÉS PÉTER";
15 PRINT"■                                        ■";
16 PRINT "■■  A PROGRAM UJ LEHETOSEGEI:"
17 PRINT "■■  -A TUZGOMBOK ES A NYILAK ATMENNEK AZ    EMBERUNKON'
18 PRINT "■  -A TOBBI SZABALY TOVABBRA IS ERVENYES"
19 PRINT"■■           ■TURELEM , DOLGOZOM !"
20 PRINT"■■■■■ ■  ■┌──────────────────────────┐■"
21 PRINT"    ■| (C) PIPROG SYSTEM HOUSE , 1984 |■"
22 PRINT"    ■└──────────────────────────┘■"
23 IF A=0 THEN A=1:LOAD"QUASI",8,1
24 PRINT "■■■■■■■■■■■■■■■■          ELKESZULTEM A TOLTESSEL ! "
25 POKE22283,0:POKE22284,192
26 POKE22342,0:POKE22343,192
27 POKE22399,0:POKE22400,192
28 POKE49152,0
29 PRINT "■             INDULHATUNK (■I■) ?";
30 GETA$:IFA$<>"I"THEN 30
31 SYS 16384
```

VC 20

RÁZÓS ÚTON

A játékos a „Z" gombbal balra, a „/" gombbal jobbra, a „C"-vel le és „,"-vel felfelé irányíthatja karakterét. Négyszer lehet akadályba ütközni, de falnak sosem! Vigyázat! Forduláskor ne ütközzünk saját nyomunkba!

A szerző csúcsteljesítménye: 60-as nehézségi fokon 276. Tessék túlszárnyalni!

PALOTAI PÉTER

```
1 Print"milyen nehez Palyat akarsz?"
   iranyitas /:jobb z:bal c:le ,:fel"
2 inPuta:Print"shft+clrh":fort=1toa
3  x%=rnd(1)*21*22
4 k=7680:Pokek+x%,81:Poke36879,14
5 nextt
6 v=1:a=1:b=0
7 9eta$:ifa$="/"then18
8 ifa$="z"then19
9  ifa$="c"then20
10 ifa$=","then21
11 x=x+a:y=y+b
12 ifx>21orx<0ory>22ory<0then25
13 ifPeek(k+x+22*y)=81
   orPeek(k+x+22*y)=90then22
14 fort=1to150:nextt
15 Pokek+x+22*y,42:Pokek+x-a+22*(y-b),90
   :Poke38400+x-a+22*(y-b),7
16 Print tab(18)c;"clrh":c=c+1:
   Poke36878,12:Poke36876,220:
   Poke36876,0
17 9oto7
18 a=1:b=0:9oto11
19 a=-1:b=0:9oto11
20 a=0:b=1:9oto11
21 a=0:b=-1:9oto11
22 Print"clrhK A M I K A Z E"v:
   Poke36877,135:Poke36877,0
23 v=v+1:ifv>4then25
24 9oto11
25 forw=128to150steP.3
26 Print"ve9e";:Poke36875,w:Poke36875,0
27 nextw
28 Printtab(66)c"lePest tettel"
29  end
```

COMMODORE 64

Játékprogram-módosítás

Először töltsük be a SUPER MONITOR programot, aztán ennek segítségével a FALCON PATROL-t, majd disassembláljuk a programot 41FF-től 4223-ig. A 4207-es címen levő STA utasítás tartalmát változtassuk meg 80-ra, és ugyancsak 80-at írjunk a 420B címen levő AND műveletbe. Ezután monitor üzemből indítsuk el a programot G 4100-zal. Most már nem robban fel sprite-unk ütközéskor.

PAPP TIBOR

Coby-vadászat

A játék lényege a következő. A megjelenő játékmezőben a kerettel párhuzamosan halad a vadász (egy fekete kör). Az ellenséges fekete négyzetek véletlenszerűen jelennek meg a játéktér bal, illetve jobb felén, attól függően, hogy a vadász éppen merre halad. A fekete négyzetek mindig akkor jelennek meg, amikor a vadász az 1. ábrán látható felső-középső pozícióba ér.

Lövés hatására a lövedék mindig az ellenség felé halad. Amennyiben célt ér, a fekete négyzet pepitává változik, és arra már nem szabad lőni. Ha véletlenül mégis eltalálnánk, eltűnik.

Az alsó ablakokban a program számolja a vadász „lépéseit" (ez mindig 1032 a játék végére), mellette jobbra a megtett köröket láthatjuk, a szélén pedig az érvényes találatokat. Az alsó sorban számolja a lövéseket, mellette az összpontszámot és végül a szélén a hibás találatokat.

A RUN parancs begépelése után a program kirajzolja a játékteret és vár, hogy a billentyűzetről bármelyik gomb benyomásával indítsuk a játékot. Amikor ez megtörtént, a vadász elindul a felső-középső pozíciójából, és megjelenik az első négy fekete négyzet. Kezdődhet a vadászat.

A billentyűzet bármelyik gombjával elindítható a lövedék. Ilyenkor három eset lehetséges:

a) Ha a lövedék nem talál, akkor −5 pont (levonás).

b) Ha a lövedék eltalál egy fekete négyzetet, akkor az megváltoztatja színét, és +23 pontot jelent.

c) Ha a lövedék egy pepita négyszöget talál el, −9 pont (levonás).

A játék 12 körig folytatódik, utána a program leáll, és a legfelső sorban kiírja az összpontszámot, ami úgy jön létre, hogy az összpontszámból kivonja a lépésszámot. A találatok esetén azért van +23 és −9 pont, hogy a számok látványosan ugráljanak, ne csak kerek értékeket mutassanak.

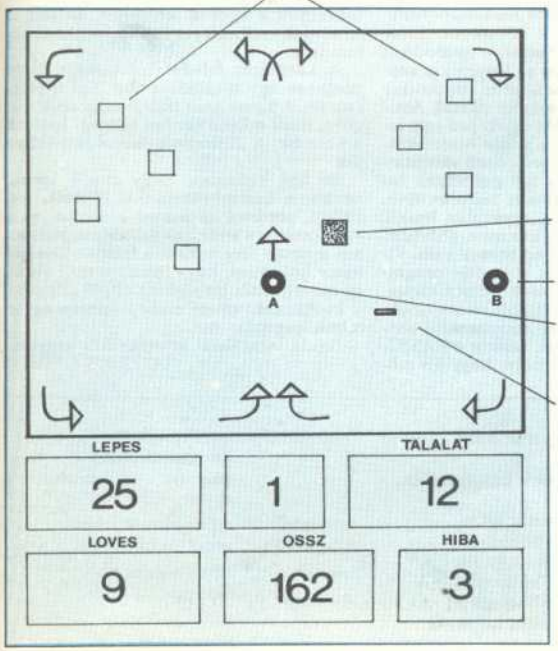
A program listájának szerkezete:

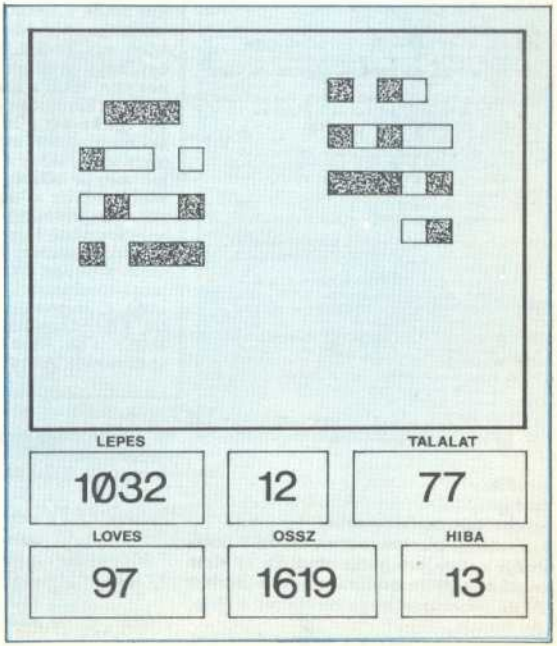
0— 20	a játéktér kirajzolása
20— 49	értékadások, körszámlálás, kiírás
50— 89	a vadász mozgása
100—110	lépés számlálása, kiírása, a vadász megjelenítése
500—540	a lövedék négy irányban történő mozgása
600—610	az érvényes találat számlálása, kiírása, lövéshang
650—660	a hibás találat számlálása, kiírása, lövéshang
700—760	a fekete négyzetek véletlenszerű megjelenítése
800	robbanáshang
900—901	összpontszám kiszámítása, kiírás, játék vége
1000—	a lövedék rajzolása

1. ábra. Véletlenszerűen megjelenő akadályok ebben a pozícióban nem szabad lőni

VC 20

```
1 rem * COBY VADASZAT*
2  Print"shft+clrh" a=7680:Poke36879,216
3 forc=0to505:Poke8400+c,0:next
8 forc=0to3*22step22:Poke8023+c,160
   :Poke8091+c,160:next

9forc=0to8*22step22:Poke8018+c,160:next
10 forc=0to21:Poke a+c,128
   Poke a+22*22+c,128:Poke a+308+c,160
   Poke a+396+c,160:next
11 forc=0to22*22step22:Poke a+c,128
   Poke a+21+c,128:next


12 Print"<clrh><rvson><14*crsrle><2*crsr↓"
   lepes        talalat"
13  Print"<rvson><3*crsrle><2*crsr↓obb>
   loves   ossz  hiba"
20 clr
21 v=36878:s1=36875:s2=36877
46 geta$:ifa$=""then46
47 a=7713:Pokev,15:gosub700:k=k+1:
   ifk>11thenPrint"<clrh><rvson><crsr↓
   obb>    utolso kor"
48 ifk>12then900
49 Print"<clrh><16*crsrle><9*crsr↓obb>"k

50 a=a+1:q=240:gosub100:gosub500:Poke a,32

52 ifa=7722then55
54  goto50
55 a=a+22:q=230:gosub100:gosub510:
   Poke a,32
56 ifa=7986then60
58 goto55
60 a=a-1:q=220:gosub100:gosub520:
   Poke a,32
62 ifa=7977then65
64 goto60
65 a=a-22:q=210:gosub100
66 ifa 7713then9osub75:9oto70
69 9oto65
70 a=a-1:q=220:9osub100:gosub500:
   Poke a,32
72 ifa=7703then75
74 goto70
75 a=a+22:q=230:gosub100:gosub530:
   Poke a,32
```





```
77 ifa=7967then80
79 goto75
80a=a+1:q=240:gosub100:gosub520:
   Pokea,32
82ifa=7977then85
84 goto80
85a=a-22:q=210:gosub100
87 ifa=7713then90to47
88 goto85
100 l=l+1:Print"{clrh}{16*crsrle}{2*
   crsrjobb}"l
102Pokea,81:Pokes1,q
103for i=0to30:next
105 Pokea,32:Pokes1,0
110return
500 geta$:ifa$=""thenreturn
501 g=901 b=a:c=a:Peka,81
502c=c+22
503 ifPeek(c)=160then600
504 ifPeek(c)=102then650
505w=66:gosub1000
506ifc=b+(12*22)then800
507 goto502
510 geta$:ifa$=""thenreturn
5129=f+1 c=a:b=a:Peka,81
514 c=c-1
515 ifPeek(c)=160thhen600
516 ifPeek(c)=102then650
517 w=67:gosub1000
518 ifc=b-19then800
519 goto514
520 geta$:ifa$=""thenreturn
522 g=g+1:c=a:b=a:Peka,81
524 c=c-22
525ifPeek(c)=160then600
526  ifPeek(c)=102then650
527 w=66:gosub1000
528 ifc=b-(12*22)then80
529 goto524
530 geta$:ifa$=""thenreturn
534 g=g+1:c=a:b=a:Peka,81
535 c=c+1
536 ifPeek(c)=160then600
537 ifPeek(c)=102thhen650
538 w=67:gosub1000
539ifc=b+19then800
540 goto535


600 Pokec,102:t=t+1:Print"{clrh}{16*
    crsrle}{15*crsrjob}"t:f=f+23
601 Print"{3*crsrle}{9*crsrjobb}"f
605 Pokes1,0:fori=15to0step-.2:Pokes2,
    235+i:Pokev,i:next:Pokes2,0:Pokev,15
610 return
611 h=0
615 Printh
650 Pokec,42:h=h+1:Print"{clrh}{20*
    crsrle}{17*crsrjobb}"h:f=f-9
651 Print"{clrh}{200*crsrle}{9*crsrjobb"
    f
655Pokes1,0:fori15to0step-.1:Pokes2,200
    +i:Pokev,i:next:Pokes2,0:Pokev,15
660 Pokec,32:return
700 x=int(5*rnd(0)+1):Poke7759+x,160:
    x=int(5*rnd(0)+1):Poke7803+x,160
702  x=int(5*rnd(0)+1):Poke7847+x,160:
    x=int(5*rnd(0)+1):Poke7889+x,160
705 return
750 x=int(5*rnd(0)+1):Poke7770+x,160:
    x=int(5*rnd(0)+1):Poke7814+x,160

755 x=int(5*rnd(0)+1):Poke7858+x,160:
    x=int(5*rnd(0)+1):Poke7902+x,160
760 return
800 Pokes1,0:fori=15to0step-.1:Pokes2,
    220+i:Pokev,i:next:Pokes2,0:Pokev,15
801 f=f-5:Print"{clrh}{20crrsrle}{9*
    crsrjob}"f:return
900 Print"{clrh}{rvs9n}{crsrjobb}ossz
    Pontszam"f-1
901 goto901
1000 Pokec,w:Pokes1,245:Print"{clrh}
     {20*crsrle}{2*crsrjobb"g
1004 Pokec,32
1005 return
```

A megoldott Tantalizer

A címben szereplő angol szó annyit jelent, hogy „tantaluszi kínokat okozó". E nem éppen hízelgő nevet egy múlt századbeli logikai játék kapta, amely különböző formákban ugyan, de újra meg újra felbukkan azóta is.

A játék alapváltozatában négy, a lapjain mintával ellátott kockából áll. A legegyszerűbb kiszínezni az oldalakat; mi az ábécé betűivel fogjuk a mintákat azonosítani. A cél egy olyan $1 \times 1 \times 4$-es torony építése a kockákból, amelynek egyes oldalain csupa különböző minta (szín) szerepel. Az *1. ábrán* a kockák színezése látható.

Az első világháborúban ugyan felbukkant egy ötkockás változat Belgium, Franciaország, Japán, Oroszország és Anglia zászlajával, kísérleteztek hat kockával is, de egyik sem vált olyan elterjedtté, mint az eredeti. Nálunk Taktikolor néven lehetett az üzletekben kapni a négykockás verziót. Csak szerkezetében különbözik tőle a Bognár-golyók nevű logikai játék, amelyben a kockák helyét golyók veszik át, melyek középpontjuk körül elforgathatóan vannak elhelyezve egy, a forgathatóság érdekében hasítékokkal ellátott, átlátszó műanyag tokban.

Elvileg $24^4 = 331\,776$-féleképpen lehet összerakni a tornyot, mivel egy kocka 24-féleképpen forgatható el, a sorrendjük viszont nem érdekes — a Bognár-golyóknál rögzített is. Ez a szám 1/8-ára csökken, ha nem tekintjük különbözőeknek a szimmetrikus állásokat.

A feladat szisztematikus megoldása tipikus példa a visszalépéses keresésre (backtracking). A módszer a következő. Lerakunk egy kockát, rátesszük a másodikat úgy, hogy egyik oldalon se legyen két azonos szín, majd a harmadikat és negyediket is ehhez hasonlóan — már ha sikerül. Amikor egy kockát a 24 állás egyikében sem lehet megengedett módon a már összerakottakra tenni, akkor a megelőzőnek változtatjuk meg a helyzetét, s így próbálunk továbblépni. Ez az algoritmus biztosan elvezet a megoldásig — ha egyáltalán létezik —, csak kissé lassan, s könnyen eltévesztheti. Egyszóval nem embernek való. Viszont kiválóan alkalmas különféle programozási módszerek, stílusok bemutatására.

Az *1. programot* — kisebb átalakításoktól, a Spectrumhoz való igazításoktól eltekintve — kb. 8 éve írtam, amikor a BASIC-et tanultam. Annak ellenére, hogy jól működik, nehéz lenne jól megírtnak minősíteni. Nincsenek benne például jól elkülönítve az egyes funkciók, nehéz áttekinteni a sok feltétel és ide-oda ugratás miatt. Hasonló, akár csak méretében eltérő feladat megoldásához az egész programszöveget át kellene bogarászni a megfelelő javítások elvégzéséhez.

A *2. programot* nemrég írtam, felvértezve a strukturált programozás eszméivel és a Beta basic 3.0 programnyelvvel. Nem célom ezek egyikét sem most részletesen ismertetni, ez a lap előző és mostani számában részben megtörtént, csak a lista érthetősége érdekében teszek néhány megjegyzést.

A DEF PROC és END PROC közötti utasítássorozat egy névvel ellátott szubrutint (ún. procedure-t) alkot, amelynek hívása nevének leírásával történik. A szubrutin neve után változók és konstansok állhatnak, ezekkel cím és érték szerinti paraméterátadást végezhetünk a szubrutin formális paraméterlistájának megfelelően. A szubrutinok egymást és saját magukat is hívhatják. A változók keveredését helyi, csak a szubrutinon belül ismert változó bevezetésével akadályozhatjuk meg. Így például a Keres szubrutin „szint", „l", „j", „k", és „lepes" változója annyi példányban létezik, amilyen mélyen egymásba ágyazódnak a hívások.

A moduláris felépítés és a feltétlenül szükségesnél kissé általánosabb program írása megkönnyíti a program áttekintését, elősegíti az esetleges későbbi módosítások könnyen elvégezhetőségét. Az egyszerűség érdekében a kockák lehetséges állásait és az aktuális színezést is DATA sorokban adtam meg.

A Tantalizer feladat érdekessége, hogy pontosan egy megoldása van. Ezt a program kb. 3-5 perc alatt találja meg, attól függően, hogy milyen kezdeti állásból indítjuk a keresést. A *2. ábrán* a megoldást láthatjuk.

Be kell vallanom, hogy annak idején, amikor a kezembe került ez a játék, gép nélkül, papíron, ceruzával oldottam meg. Kihasználva a színek eloszlását, meglehetősen gyorsan megoldható a feladat. Érdekes lenne kideríteni, hogy létezik-e még olyan, az ismertetettől lényegesen eltérő színezése a kockáknak, amely mellett szintén egyértelmű megoldás van.

Ennek vizsgálatát az olvasókra hagyom.

LOVRICS LÁSZLÓ

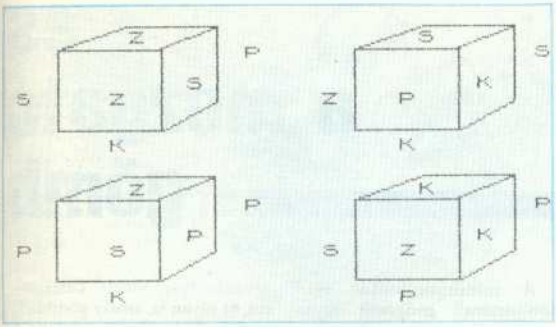
A szerkesztőség megjegyzése a két VC20-as programhoz:

A listában szereplő jelek a következőket jelentik:

"crsrle"	a fénypontot lefelé mozgató billentyű
"shft + crsrle"	a fénypontot lefelé mozgató billentyű shift gombbal együtt
"clrh"	a fénypontot bal felső sarokba mozgató billentyű
"shft + clrh"	a fénypontot bal felső sarokba mozgató billentyű shift gombbal együtt
"crsrjobb"	a fénypontot jobbra mozgató billentyű
"rvson"	háttérszínváltó, bekapcsolás
"rvsoff"	háttérszínváltó, kikapcsolás

lövedék
vadász
egyszer már eltalált akadály
2. ábra. Egy közepes szinten lejátszott játék ábrája a játék befejezése után. Ez az ábra mindaddig látható marad, míg a RUN/STOP billentyűvel meg nem állítjuk a program futását.

NAGY LÁSZLÓ ISTVÁN



1. ábra

```
1.   megoldas:

1.   KZZP    SS
2.   SPKS    ZK
3.   PKSZ    PP
4.   ZSPK    KP

1 megoldas volt.
```

2. ábra

1. program

```
 10 DEF FN w()=PEEK 23672+256*(PEEK 23673+256*PEEK 23674)
 20 DIM a(24,6)
 30 FOR k=1 TO 24
      FOR l=1 TO 6
        READ a(k,l)
      NEXT l
    NEXT k
 40 READ n
    DIM t$(n,6)
    DIM h(n)
 50 FOR k=1 TO n
      LET h(k)=1
      READ t$(k)
    NEXT k
 60 LET w=FN w()
 70 LET r=0
    LET j=1
 80 FOR k=1 TO j-1
      FOR l=1 TO 4
        IF t$(k,a(h(k),l))=t$(j,a(h(j),l)) THEN GO TO 210
 90   NEXT l
    NEXT k
100 LET j=j+1
110 IF j<=n THEN GO TO 80
120 LET j=j-1
130 LET r=r+1
140 CLS
    PRINT r;".  megoldas:"
    PRINT
150 FOR k=1 TO n
      PRINT
160   PRINT k;".  ";
170   FOR l=1 TO 4
        PRINT t$(k,a(h(k),l));
      NEXT l
      PRINT "   ";
180   FOR l=5 TO 6
        PRINT t$(k,a(h(k),l));
      NEXT l

190 NEXT k
200 REM PAUSE 0
210 LET h(j)=h(j)+1
220 IF j=1 THEN LET h(j)=h(j)+7
230 IF h(j)<=24 THEN GO TO 80
240 LET h(j)=1
    LET j=j-1
250 IF j>0 THEN GO TO 210
260 CLS
    PRINT r;" megoldas volt."
270 PRINT (FN w()-w)/50;" sec."
280 STOP
290 DATA 2,3,5,4,1,6,2,4,5,3,6,1
300 DATA 3,5,4,2,1,6,4,5,3,2,6,1
310 DATA 5,4,2,3,1,6,5,3,2,4,6,1
320 DATA 4,2,3,5,1,6,3,2,4,5,6,1
330 DATA 1,4,6,3,2,5,1,3,6,4,5,2
340 DATA 4,6,3,1,2,5,3,6,4,1,5,2
350 DATA 6,3,1,4,2,5,6,4,1,3,5,2
360 DATA 3,1,4,6,2,5,4,1,3,6,5,2
370 DATA 1,2,6,5,3,4,1,5,6,2,4,3
380 DATA 2,6,5,1,3,4,5,6,2,1,4,3
390 DATA 6,5,1,2,3,4,6,2,1,5,4,3
400 DATA 5,1,2,6,3,4,2,1,5,6,4,3
410 DATA 4
420 DATA "KZSSPZ"
430 DATA "KPKZSS"
440 DATA "KSPPPZ"
450 DATA "PZKSPK"
```

2. program

```
 10 DEF PROC Tantalizer
 20  LOCAL kockaszam,oldalszam,allasszam,vizsgalt,megoldas,ugras,h(),a(),t$
 30  LET megoldas=0
 40  Olvas
 50  Keres kockaszam
 60  PRINT ''megoldas;" megoldas volt."
 70 END PROC
 80 DEF PROC Keres szint
 90  LOCAL j,k,l,lepes
100  IF szint=0 THEN Kiiras
      GO TO 200
     ELSE IF szint=1 THEN LET lepes=ugras
     ELSE LET lepes=1
110  FOR k=1 TO allasszam STEP lepes
120   FOR j=szint+1 TO kockaszam
130    FOR l=1 TO vizsgalt
140     IF t$(szint,a(k,l))=t$(j,a(h(j),l)) THEN GO TO 190
150    NEXT l
160   NEXT j
170   LET h(szint)=k
180   Keres szint-1
190  NEXT k
200 END PROC
210 DEF PROC Kiiras
220  LOCAL k,j,l
230  LET megoldas=megoldas+1
240  CLS
     PRINT megoldas;".  megoldas:"
     PRINT
250  FOR k=1 TO kockaszam
      PRINT
260   PRINT k;".  ";
270   FOR l=1 TO vizsgalt
       PRINT t$(k,a(h(k),l));
      NEXT l
      PRINT "   ";
280   FOR l=vizsgalt+1 TO oldalszam
       PRINT t$(k,a(h(k),l));
      NEXT l
290  NEXT k
300 END PROC
310 DEF PROC Olvas
320  LOCAL j,k,l
330  RESTORE 410
340  READ allasszam,oldalszam
350  DIM a(allasszam,oldalszam)
360  FOR k=1 TO allasszam
      FOR l=1 TO oldalszam
       READ a(k,l)
      NEXT l
     NEXT k
370  READ kockaszam
```

```
    DIM t$(kockaszam,oldalszam)
    DIM h(kockaszam)
380 FOR k=1 TO kockaszam
     READ t$(k)
    NEXT k
390 READ vizsgalt
    READ ugras
400 END PROC
410 DATA 24,6
420 DATA 2,3,5,4,1,6,2,4,5,3,6,1
430 DATA 3,5,4,2,1,6,4,5,3,2,6,1
440 DATA 5,4,2,3,1,6,5,3,2,4,6,1
450 DATA 4,2,3,5,1,6,3,2,4,5,6,1
460 DATA 1,4,6,3,2,5,1,3,6,4,5,2
470 DATA 4,6,3,1,2,5,3,6,4,1,5,2
480 DATA 6,3,1,4,2,5,6,4,1,3,5,2
490 DATA 3,1,4,6,2,5,4,1,3,6,5,2
500 DATA 1,2,6,5,3,4,1,5,6,2,4,3
510 DATA 2,6,5,1,3,4,5,6,2,1,4,3
520 DATA 6,5,1,2,3,4,6,2,1,5,4,3
530 DATA 5,1,2,6,3,4,2,1,5,6,4,3
540 DATA 4
550 DATA "KZSSPZ"
560 DATA "KPKZSS"
570 DATA "KSPPPZ"
580 DATA "PZKSPK"
590 DATA 4
600 DATA 8
```

PRO-KONTRA GM.
1074 Budapest,
Csengery u. 7. fszt. 1/a
☎ 417-893

Cégünk ajánlataiból:

— **64 k-s memóriabővítő Commodore C16-hoz**
— **FAST—VC—1541 kommunikációgyorsító rendszer a C64-hez (minden gép—floppy közötti és floppyn belüli műveletet 4—12-szeresére gyorsít)**

Konfiguráció beszereléssel együtt 7000,— Ft

— **Abszolút programvédelem C64-hez: cartridge-ben, műgyantával kiöntött EPROM-ba égetéssel, MÁSOLHATATLANNÁ teszi programjait**
— **IEC buszról vezérelhető méréspontváltó egység 2×30 vagy 1×60 bemeneti, 2, ill. 4 kimeneti csatornával**
— **Digitális kijelzésű elektornikus óra 8×14 cm-es digitmérettel, különféle színekben**
— **Egyéb, közepes sorozatú fejlesztések igény szerint**

Pro-Kontra Automatizálási,
Műszaki Tanácsadó és Közvetítő GM

Budapest, Csengery u. 7. fsz. 1/a 1074
Tel.: 417-893
Levélcím: Budapest, Pf. 72. 1581
Telex: 22-7770

Az egér és a mozgató gömb

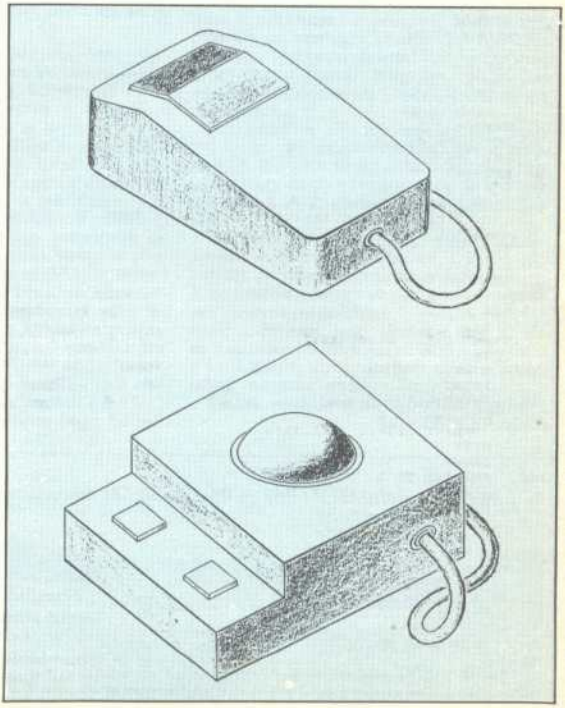
A számítástechnikai egér történetének gyökereit egyes szerzők egy múlt századi játékig, mások csak a hatvanas évek elejéig, D. Engelbart (Stanford Research Institute) munkásságáig vezetik vissza. Mivel nem áll rendelkezésünkre kellő mennyiségű és minőségű technikatörténeti adat, ezzel az érdekes kérdéssel nem tudunk érdemben foglalkozni, csupán néhány történeti vonatkozású mejegyzést teszünk.

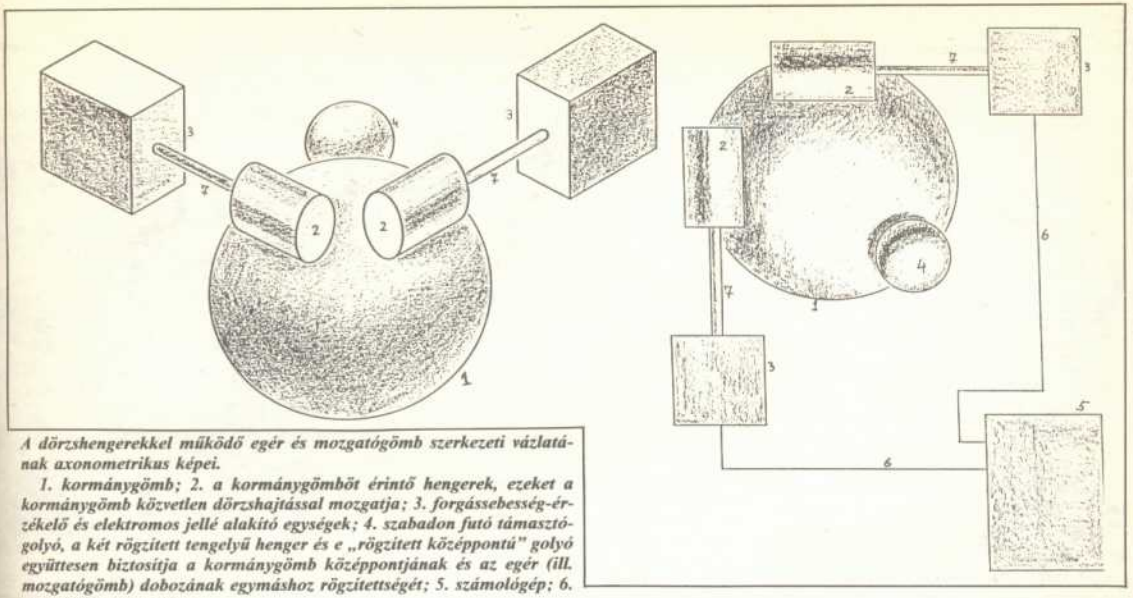
Először is nem beszélhetünk az egérről, mint egységes működési elvű eszközről. Van ugyanis egér, amely kerekekkel, és olyan is, amely gömbbel (golyóval) érintkezik a talajjal. A kerekek, illetve a gömb mozgását is többféleképpen érzékelik és dolgozzák fel az egyes egérkonstrukciók.

A másik kétségtelen tény az, hogy az előzők alapján nem egértörténetről, hanem egérfajták történetéről kell beszélnünk.

Tény továbbá, hogy egy egérből (nem *az* egérből!) az Apple csinált világsztárt, és ez az egér a reklámcsinálta, nagyra felfújt sztárok tömegében valóban megérdemelte mind-

Az egér, ill. mozgatógömb axonometrikus rajzai





A dörzshengerekkel működő egér és mozgatógömb szerkezeti vázlatának axonometrikus képei.

1. kormánygömb; 2. a kormánygömböt érintő hengerek, ezeket a kormánygömb közvetlen dörzshajtással mozgatja; 3. forgássebesség-érzékelő és elektromos jellé alakító egységek; 4. szabadon futó támasztógolyó, a két rögzített tengelyű henger és e „rögzített középpontú" golyó együttesen biztosítja a kormánygömb középpontjának és az egér (ill. mozgatógömb) dobozának egymáshoz rögzítettségét; 5. számológép; 6. vezetékek; 7. tengelyek.

azt a szakmai és anyagi sikert, amit eddig learatott.

Végül pedig az is tény, hogy nem az Apple volt az első, amely egér nevű perifériális egységet kapcsolt számológéphez.

Douglas Engelbart villamosmérnök a billentyűzetről való utasításadás nehézségét kívánta megszüntetni a képernyőn való mozgási, rajzolási tevékenységek esetében. A hagyományos billentyűzet nyilvánvalóan jó az alfanumerikus információbevitelnél, de kényelmetlen például egy pályavonalnak a képernyőre juttatásánál. Az eszköz, amit szerkesztett, lényegében mechanikai perifériális egység volt. Külső mozgást, illetve helyzetet alakított át gépen belüli és képernyőn való mozgássá, illetve helyzetté.

A *pointing device*, azaz mutatókészülék névvel illetett vezérlőeszközre — külleme miatt — hamarosan ráragadt az egér név, és az így is vált napjainkban közismertté.

Elgondolkodtató az egér szokatlanul nagy sikere. Ennek okát abban kereshetjük, hogy ezzel mozgást vele hasonló mozgással vezérelhetünk a gépben, illetve a képernyőn. Alkalmazásával nem kényszerülünk absztrakt közbülső műveletek elvégzésére.

Ezenkívül az Apple cég Lisa és Macintosh gépein az egérnek a mozgatásban mutatkozó előnyeit szoftver segítségével még a gazdag szolgáltatáskínálatból való választás terén is kamatoztatták. Az egér sikere enélkül talán nem lett volna olyan nagy és átütő.

Visszatérve az egerek nyüszáki megoldására, meg kell említenünk azt az egérfajtát, amelyben sem kereknek, sem gömbnek nincs lényeges szerepe, ugyanis az elmozdulást nem ezek elmozdulása közvetíti a szerkezetbe, hanem optikai jelek, amelyeket speciális hálós papír vonalainak az egér érzékelői alatt való elhaladása gerjeszt.

Az egereken gomb is van, néha egy, néha több. Legalább egy gombnak azért kell lennie, hogy legyen mivel közölni a géppel a „most" vagy „itt" vagy a „készen vagyunk" információt. Mivel információbejuttatási eszközről van szó, szükség van arra, hogy közölni lehessen a géppel, hogy egy pillanatnyi helyzettel valami mást is kell csinálni, mint a neki megfelelő képpont képernyőn való megjelenítését vagy világító állapotban tartását.

Pusztán az elvek tisztázása érdekében képzeljük el, hogy a képernyő 1:1 arányú rajza ott van mellettünk az asztalon. A fénypont (cursor) annak a pontnak megfelelő ponton világít, ahol egerünk gömbje a rajz lapját érinti. Egerünket — ha olyan típusú, hogy ez előírás — önmagával párhuzamosan mozgatva a papíron azt tapasztaljuk, hogy a képpont mozgása utánozza az egérét. Ha az egér 1 centimétert halad előre, a fénypont ugyanannyit tesz meg felfelé. Ha az egér balra mozog, a képpont is azt teszi. Ha az egér leír egy parabolaívet, a képpont is parabolaíven mozog. (Rajtunk múlik, hogy a befutott pályát világító állapotban kívánjuk-e hagyni, vagy sem.) Ha az egér visszatér kiindulási pontjára, a képpont is. Ez nyilvánvalóan a rendszer pontosságától is függ. A „visszaállási pontosság" természetesen egy egérnél sem éri el a rajzológépek hasonló adatát.

E rövid leírásból is jól érzékelhető, hogy az egeret kézben tartva azt érezzük — joggal —, hogy a képernyőn levő fénypontot tartjuk kézben, és az egeret mozgatva a fénypontot mozgathatjuk közvetlen módon. Ennek a lélektani előnynek a kiaknázása ösztökélte és ösztökéli a kutatókat az egerek és egérszerű perifériális egységek további fejlesztésére.

Befejezésül egy olyan „egérrokont" mutatunk be, amelynek előtörténetéről nem sokat tudunk. Felfedezéséhez valószínű, hogy egy felfordult (nem megdöglött!) vagy felfordított egér segítette hozzá megalkotóját, de az is lehet, hogy ez előzte meg az egeret.

A „forradalmian" új szerkezet a *tracker ball* vagy *track ball* nevet kapta, és az egérhez hasonlóan helyet kapott az Apple és más gépek bemeneti egységei között.

A track ball semmi más, mint egy hátán fekvő egér, amelynek a gömbjét kézzel lehet mozgatni. Magyar nevéül *mozgatógömb*öt javaslunk, mert vele a fénypontot mozgathatjuk. A botkormány mintájára alkotott *gömbkormány* kifejezést sem tartjuk rossznak.

Ez az új információbejutattó eszköz az egérhez hasonlóan meghagyja azt a lelki előnyt, hogy kezünk mozgásának közvetlen, egyszerű és természetes mozgáskövetkezményét látjuk a képernyőn, használatának helyigénye azonban kisebb, mint az egéré. Hogy a pontossága milyen, nem tudjuk. Még nem volt szerencsénk vele méréseket végezni. Véleményünk szerint azonban ezeknek az eszközöknek a pontosságát nem is értelmes dolog a kézmozgatás pontosságát meghaladó mértékűre fejleszteni. Ezenkívül pedig a pontosság fokozásának vannak más — például szoftverrel elvégeztethető — útjai is.

A pontosság nyilvánvaló korlátja ma a mozgatási sebességgel kapcsolatos. Az egérfélék ugyanis jelenleg (még?) nem bírják a „nagy" sebességeket. 1 km/óra (kb. 25 cm/s) felett leképező tevékenységük hasznavehetetlenné válik.

SZEBENSZKI SÁNDOR

Nyári árak Bécsben

A Bécsben 1986. június első hetében érvényes árakból sorolok fel néhányat. Annyit azonban hozzátennék, hogy Mexico-Platz környéki jugoszláv, török származású üzlettulajdonosok boltjain kívül a városnak más, a magyarok által kevéssé ismert üzletnegyedei is vannak, ahol a turisták rendkívül kulturált körülmények között, a Mariahilfer Strassehoz viszonyítva jóval olcsóbban és jobb minőségben szerezhetik be kézzelfogható „úti élményeiket".

Sajnos elterjedt hiedelem, hogy a városon kívüli diszkontáruházak is sokkal olcsóbbak. Lehet, hogy más cikkeknél ez igaz, de a számítógép és tartozékai tekintetében úgy tapasztaltam, hogy áraik megegyeznek a legnagyobb belvárosi áruházak viszonylag magas árszintjével.

Aki szeret bütykölni, ha számítógépet nem is, de sok remek tartozékot egységcsomagban, ún. Bausatzban kb. a kész készülék árának egyharmadáért vásárolhat a PRINT—TECHNIK Bécs, Stumpergasse 34. szám alatti üzletében. Itt vehet a Commodore—64-es legfontosabb speciális IC-iből:

Típus-jelzés	Ár (schilling)	
6522	153	perifériaillesztő VC—20, VC—1541
6502	139	mikroprocesszor VC—20, VC—1541
6510	450	mikroprocesszor C—64
6526	460	perifériaillesztő C—64, VC—1570/1571
6581	672	szintetizátor C—64, C—610
EPROMOK		
2764	75	
27128	79	

Gépek és kiegészítők

A belvárosban a legolcsóbbak az ÖTLET 1986. márciusi számában a „Magyarosch vásárlásch" c. cikkben említett üzletek, a Mariahilferstrasséből nyíló Esterházygassén levő Sylvia és Audio-Center. Az árak a forgalmi adóval terhelt összegek.

C—64: 3150—3900 schilling. A forgalmi adó (MWST) 20%, tehát a tényleges költség a visszatérítés után 2600—2800 schilling.

C—16: még csak a gyengébben forgalmazó üzletekben van, ára 1450—1700 schilling.

C64—II: Doboza a C—128-éra emlékeztet. Még csak mutatóban volt, ezért csak az akkor még várható árat tudom: 4500 schilling.

C—128: 5600—7990 schilling, ez utóbbinál néhány százzal drágábban már a 128 D változat is kapható, a billentyűzet alatt elhelyezett floppyval.

A Sinclair ZX-Spectrum 48 k csillaga leáldozóban van, már csak kevés helyen tartják, 1450—1990 schillinges áron.

ATARI 800 XL. Új árháború kezdett kibontakozni. Ezt az alapgépet, amely vetekszik a C—64-gyel, az 1050 jelzésű, 3,5 collos hajlékonylemez-meghajtóval együtt 3580 schillingért, akciós áron kínálják.

A VC—20 alapgép 990 schilling, de már csak a Flohmarkton (bolhapiac). VC—20 alapgép garancia, tápegység, antennaillesztő nélkül, kifejezetten amatőr célokra a PRINT—TECHNIK-nél 499 schilling. Ha azt vesszük, hogy a C—16-tal szemben ennek van user portja, ilyen áron már megkísérelném rábízni a modellvasút irányítását. Ha valamit netán elrontanék, mindössze 1500 forintom bánná.

A VC—1541 lemezmeghajtó ugyancsak a bukó ágon van. Lassan kifogy az üzletekből, de árát tartja: 3600—4250 schilling.

A VC—1570 és kétoldalasan dolgozó testvére, az 1571-es már a C—128-cal való együttműködésre készült. Ára 3850—5600 schilling. Az 1570-es doboza az 1541-essel azonos, míg az 1571-es laposabb, kissé szélesebb. Érdekességük, hogy két formátumban is dolgoznak. Az ún. GCR formátum teljesen azonos az 1541-esével, az MFM formátum pedig lehetővé teszi egy lemezoldalon max. 200 k tárolását. Ebben az üzemmódban a blokkokat nem 256 bájtokból, hanem ennek feléből vagy egész számú többszöröséből kialakítva írja lemezre. Valószínűleg így érték el, hogy az azonosítást, ellenőrzést szolgáló bájtokat egyszer felírva, fizikailag több hely maradt a lényegi információk rögzítésére egy hosszabb blokkban.

A demolemezen újabb rendszerprogramok is vannak, melyek például a Shell-DOS C—128-cal való együttműködését támogatják. A C—128-cal való illeszkedésre utal kézikönyvének az a megjegyzése, hogy képes gyorsabb működésre is attól függően, hogy a C—128 milyen üzemmódban

dolgozik (C—64, C—128 vagy CP/M üzem). ROM-ja 32 k-s — szemben az 1541-es 16 k-jával. A RAM kapacitása azonos: 2 k. Soros porton át működik.

Nyomtatók

Igen széles választékban kaphatók: a Brother HR—5 1990 schilling, az MPS 801 és 803 2600—3900 schilling között.

Érdekes, hogy az iratok, levelek írására készült, de lényegében számítógépes belsővel épült írógépek C—64-hez való illesztése többnyire megoldott. Például a CANON S—50 2990 schillingért rendelkezik a magyar ábécé betűivel is, választható írásképpel. Élmény egy vele írt lapot olvasni. A Foto-Quelle eladóterébe kitett ilyen géppel volt alkalmam játszani, és nem szólt rám az eladó!

Botkormányok

A hagyományosakat többnyire már leárazták: a Quick-shot II. 120—270 schillingért, a Quick-shot I. 89—170 schillingért kapható.

Érdekes volt 499 schillingért a Herlango szakáruházban az infra-joystick. A C—64-hez a szokványos Canon-csatlakozóval illeszkedő, vevőáramkört tartalmazó dobozból és a játékkar talpát képező, az adót magába foglaló egységből áll, mellyel a drót nélküli vezérlés kényelmesen, nagy távolságból végezhető. Milyen jól védhetnénk vele játszó gyermekeink szemét a közelről figyelt képernyő ártó hatásaitól!

Hajlékony lemezek

Nagy választékban, igen eltérő árakon és minőségben, 10 darabos dobozonként 149—399 schillingért árulják.

Datasette

5—600 schilling közötti áron kapható. A C-2-N Commodore-egység helyett a 1531-es alapesetben a C—16-hoz és +4-hez illeszkedő magnót adapterrel árusítják, így alkalmas a VC—20-hoz, C—64-hez is.

A nagyobb számítógéprendszereket nem említem, hiszen a behozatali korlátozások és kiutazásaink erre nem alkalmas valutakerete miatt nem valószínű, hogy egyhamar a házi számítógépes sarok darabjaivá válnak. Az biztos, hogy vannak még e világnak csodái.

LUKÁCS ATTILA

Úgy illett volna, hogy az év utolsó számában a rovat szerkesztője summázza az olvasók véleményét, javaslatait, tanácsait. Meg sem kísérelem, hiszen nemegyszer ugyanabban a postában találok olyan levelet, ami az egyik írásunkat dicséri, míg a másik ugyanazt alaposan elmarasztalja. Vannak állandóan visszatérő problémák is, például — főleg a legifjabb olvasóink — sorozatosan követelik a reklámok megszüntetését és az így felszabaduló oldalakon „sokkal hasznosabb" programok közlését. Nem beszélve azokról, akik azt reklamálják, hogy kevés a C 64-ről szóló írás, míg mások a μM-ban is egyre jobban „burjánzó" C 64 programokat sokallják, és úgy érzik, hogy igazságtalanul megrövidítjük a Sinclair, HT, Primo híreket. Nincs egység a programnyelvek területén sem. Egyre többen szeretnék, ha nemcsak ASS és BASIC-nyelven írt programokat közölnénk, hanem más programozási nyelvekkel is megismertetnénk az olvasókat. Így születtek azután a FORTH-ról szóló írások, illetve az „Olvastuk . . ." sorozat is, amelynek szerzője — nem titkolja — PASCAL-hívő. Ha minden olvasói kívánságot nem is tudtunk teljesíteni, az olvasói leveleket komolyan vesszük, az abban leírt javaslatokat, tanácsokat a lap arculatának alakításakor figyelembe vesszük.

Pintér Sándor, Siófok,

Batthyány L. u. 27. 8600

Negyedik osztályos gimnazista, a lap rendszeres olvasója vagyok. Már három éve foglalkozom programozással, így régi vágyam teljesült, amikor idén nyáron kaptam egy Amstrad CPC 464-es számítógépet. A gép csodálatos, a gépkönyvről azonban enyhe túlzás lenne még valami hasonlót is állítani. A szerzők nem terhelték a boldog tulajdonost memóriatérképpel és más ilyen haszontalan dolgokkal, ami nélkül az assembly nyelvű programozás kissé reménytelen vállalkozás.

Így hát szívesen felvenném a kapcsolatot Amstrad tulajdonosokkal levélben vagy személyesen.

Kérem írják meg, létezik-e valamilyen nyelvű kiadás Amstardra, hol lehet ezt megszerezni; létezik-e Amstrad-klub Magyarországon?

Végül még egy kérés: írják meg kérem az Amstrad angliai címét!

Legjobb tudomásom szerint a Sinclair-klubban jönnek össze az Amstrad-gépek tulajdonosai, így ott talán társakat talál. Sajnos, az Amstrad angliai címét nem ismerjük, azt tanácsolom, írjon arra a címre, amely a gépkönyvben található.

Bartos Gyula, Budapest

Széher út 60. 1021

Engem régóta érdekel az újság, de sajnos sokat csalódom benne!

Például, hogy ne menjek messze, az 1986. júliusi számban az újságban a 27., 28., 29., 32., 33., 42., 43. oldal mind reklám.

Na most ez csak azt érdekli, aki feladja, aki közli és néha-néha valaki elolvassa. Az újság így semmit sem ér!

Megállapodást javaslok, az újságban vállalatok csak a címlap belsejében, a zárólap mindkét oldalán hirdessenek!! Pl. 1986. augusztusi címlap nagyon ötletes, kicsiszerű az alufólia, de rosszabb soha ne legyen!

Ami a reklámokat illeti, kedves Bartos Gyula, arra kérem, olvassa el a szerkesztőségi cikket és néhány korábbi számban az erről szóló írásokat.

ifj. Tomka Miklós, Budapest,

Várvíz u. 4. 1171

Már régóta tervezem ezen levél megírását. A legfrissebb (szeptemberi) Mikroszámítógép Magazinban olvastam egy érdekes ötletet, ami miatt most már nem halaszthatom tovább az írást.

Boldog C-64 tulajdonos vagyok, és egy meglehetősen nagy programgyűjteménnyel is van. (Mintegy 400 játék-, valamint 200 egyéb program: 6 programnyelv, nyelvtanító programok stb. lemezen és kazettán.) Mivel beszélek angolul és németül, néhány hónapja jelentkeztem több nyugati számítógépes újság programcsere-hirdetésére. Válaszokat is kaptam, de akkor még nem volt ekkora a programgyűjteményem, így nem voltam megfelelő cserepartner.

Itt jött ekkor az első ötlet: kellene csinálni egy C-64-klubot, mely csak programcserével foglalkozik. Találtam sok érdeklődőt, de nem eleget, így még most is újabb klubtagokat keresek. Ezért arra kérném Önöket, hogy amennyiben lehetséges, legyenek szívesek az „Az olvasó írja" rovatban ezt a levelet (vagy ennek egy részét) leközölni, hátha vannak még érdeklődők. (Néhány részlet a klubról: tagsági díj nincs, kapcsolattartás vidékiekkel postán, pestiekkel személyesen, a programcsere nem egy programot egy programért alapon megy, hanem ha valakinek kell valami, vigye, ha van valamije, akkor hozza.)

Távlati terveim is vannak: ha sikerül egy valamivel nagyobb gyűjteményt létrehozni, akkor írnék az összes általam ismert angol, német, francia számítógépes újságnak, hogy itt egy magyar C-64 Programcsereklub, mely külföldi tagokat keres. Feltehetőleg lesz érdeklődő, és akkor sok új nyugati programhoz közvetlenül hozzá lehetne jutni.

Mint ilyen klub — ha jól beindul — aztán talán vállalhatnánk egy Mikromagazin „C-64 Program" rovat vezetését, ha addig nincs más jelentkező.

Na jó, ezzel most be is fejezem, remélve, hogy tetszik az ötletem, és hogy a levelem (vagy a levelem egy részét) viszontlátom valamelyik következő Mikromagazinban.

Íme a teljes levél, közöltük. Én azt hiszem, hogy egy új Commodore-klubot nem célszerű létrehozni, hiszen az NJSzT HCC keretében már működik egy ilyen szekció. Szétaprózni az erőket célszerűtlen, inkább azt javasolom, keresse meg Őket, bizonyára örülni fognak, ha a javaslatát megvalósítja. Dr. Simonyi Endrének, a HCC vezetőjének a telefonszáma: 556-245.

Pfening András, Ürkút,

Csokonai u. 5. 8409

Az augusztusi számban olvastam, hogy a beküldött programokat programlista formájában kérik. Van néhány programom, amelyeket szívesen elküldenék Önöknek, de mivel PRIMO-ra készültek, nincs lehetőségem kiprintelni őket. Elfogadnák úgy is, ha mágneskazettán küldeném el, vagy inkább írjam le őket? Így esetleg hiba csúszhat bele. A programok grafikai jellegűek, könnyen átvihetők SPECTRUM-ra is.

A lap hasábjain dúló kritikai háborúhoz szeretnék még hozzászólni. Jó kritikára szükség van, mivel így tudjuk megállapítani a dolgok értékét. Viszont a jó kritika ismérve a jóindulat, a segítőkészség, nem pedig az áskálódás. Ezeket a felsőbbrendűséget bizonygató gúnyos szellemességeket hagyjuk meg az Élet és Irodalomnak. *Egyetértek Kovács Győző* frappáns válaszával, még időben jött, mielőtt az indulatok teljesen elszabadultak volna.

Köszönöm az egyetértő véleményt. Ami a beküldésre szánt programokat illeti, legkevésbé a kézzel írt programlistának örülünk, az ilyen programokat gyakorlatilag nem tudjuk közölni, t. i. egy alkalmas gépbe be kellene billentyűzni, azután kipróbálni (többnyire nem megy), majd kilistázni, és csak így kerülhet a lapba. Azt kérjük, hogy a programokat olvasóink mágneses adathordozón *(kazetta, floppy) küldjék és mellékeljék a kinyomtatott* forráslistát. *Egy sorba legfeljebb* 40 karaktert *írjanak, ezeket a listákat lehet ugyanis a legjobban a lapban elhelyezni. Ügyeljenek arra, hogy a printer szép világos karakterekeet nyomtasson. A leporelló* törésvonalára ne *kerüljön sor. A lapokat hajtogatás nélkül tegyék borítékba. A programhoz mellékeljenek* programleírást, *ami ne legyen szűkszavú, de ne legyen* „locsogó" *sem, annyit írjanak le, amennyi a program betöltéséhez, futtatásához, használatához szükséges.*

Ha a programhoz forráslistát nem kapunk, de a kazettát vagy a floppy-t elküldik, akkor többnyire módunkban áll a listát kinyomtatni. A mágneses adathordozót, ha kérik, visszaküldjük.

Kiss János, Nyíregyháza,

Korányi F. u. 60. II. 9. 4400

III. osztályos gimnáziumi tanuló vagyok. Számítástechnikával az év eleje óta foglalkozom, többnyire szoftver szinten. Letettem a TV—BASIC vizsgát. A számítástechnika szakterületei közül főleg az alkalmazástechnika érdekel, a számítógép és a humán tudományok közelebb hozása. Az utóbbi hónapokban átnéztem az elmúlt 1-2 év hazai számítástechnikai sajtóját, és úgy tapasztaltam, hogy a µM minden szempontból kiemelkedik közülük. A lap céltudatosan ragaszkodik megfogalmazott céljaihoz, többségében nívós cikkek jelennek meg benne, és az egyes témakörök helyes mennyiségi arányait is eltalálja.

Az ilyen magazinnak szerintem — és úgy tűnik, hogy ezt Önök is hasonlóképpen gondolják — az az egyik alapvető funkciója, hogy korrektül tájékoztassa olvasóit a számítástechnika hazai és külföldi híreiről, új információkat adjon, ezáltal egyértelművé téve bizonyos kérdéseket az olvasókban. (Ezért tetszik pl. a „Piac", a „Termékismertető", vagy a riportok ismert számítástechnikai személyiségekkel.) Másik feladata — a nálunk legelterjedtebb gépekre készült példaprogramok, szubrutinok közlésével is segítve ezt —, hogy a hobbyprogramozókat segítse munkájukban, mind konkrét példákkal, mind a programozás helyes módszerének megismertetésével (pl. Programozástechnika, Iskola-számítógép). Viszonylag kevés komplett, „konyhakész" programot kell közölni, az ilyenek terjedelme ugyanis többnyire meghaladja egy magazin kereteit.

Nagyon helyesnek tartom azt a törekvésüket, hogy ha lehetséges, a hazai gépekre készült programok közlését részesítik előnyben a külföldi gyártású, gazdag szoftvervválasztékkal rendelkező gépekkel — C-64, Spectrum stb. — szemben, hiszen azokra bárki szerezhet viszonylag könnyedén magasszintű, többnyire gépi kódban írt szoftvereket. A lap legutóbbi számaiban azonban mintha egy kicsit túlsúlyba kerültek volna a programok a többi téma rovására. Tudom, hogy sokan vannak, akik főleg a programok miatt vásárolják a lapot, tudom azt is, hogy szerették volna az új iskola-számítógépek indulását programokkal is segíteni. A régebbi számokban talán helyesebb volt az aránya a két témának. Nem hinném azonban, hogy ne találnának kellően érdekes új témákat, vagy megfelelő szerzőket. Sőt akár régi rovataik közül is felújíthatnának néhányat: nagyon tetszett pl. a régebbi számokban az „Agyafúrmány", a „Rövid és ravasz programok", vagy a „Favágás". Biztosan sokakat érdekel az augusztusi számban már elkezdődött „Fórum" rovat is stb. Még egy javaslat: szerintem is nagyon hasznos lenne egy évvégi címjegyzék az idei számokról.

Íme, egy kritikius vélemény a lapról. Azt hiszem, a lapkészítés kulcskérdése a programok, a piaci információk és a különféle leírások közötti helyes arány kialakítása, örülünk, hogy véleménye szerint a feladatot néha sikerült megoldani.

Lukács Attila, Salgótarján,

Alkotmány u. 1. 3100

Amennyiben érdekli Önöket, van néhány hardveres ötletem, amit szívesen továbbadnék, miután a gyakorlati kísérleteit sikerrel befejeztem. (A számítógépben elektronikai előéletem miatt egy kicsit bátrabban turkálok, mint más.)

Tapasztaltam, hogy a hardverhez sokan nem mernek nyúlni, ezért a mikroklubokban is sötétség van e vonatkozásban, holott Magyarországon a rádiótechnikának és az elektronikának — aminek természetes rokona a számítógépes technika — nagy múltú és széles körű amatőrtábora van. Nézetem szerint robbanásszerű változás lesz ezen a téren, amikor a szükséges alkatrészek a kereskedelemben hozzáférhetők lesznek, és a jelenleginél alacsonyabb árszinten. Ekkor fog bekövetkezni az, hogy nem lesznek majd elájulva a Commodore képességeitől, mert sokan mondják „ezt én is megcsinálom, csak jobban!"

Szívesen venném, ha a lapban a C-64-hez illeszkedő modem és EPROM-égető leírását, kapcsolási, vagy nyomtatott áramköri rajzát közölnék. A Rádiótechnika c. lap a Spectrumra é: a ZX 81-re már közölt EPROM-égetőt, dehát az más. Az is tökéletes lenne, ha ilyen kiegészítővel rendelkező kollégával összehoznának.

Örülök a lapban indult hardveres rovatnak, bár RESET-kapcsolóból ez már a második. (Az előző az ÖTLET—BITLET-ben volt.) Nehogy ezután az következzen, mint a 64'er című lap 1986. augusztusi számában, hogy hogyan kell a biztosítékot kicserélni. (Persze igaz, hogy valahol ott kezdődik a hardver.)

Egyetértek az írásával. Számítógép-építés ügyben elsősorban ötleteket keresünk és közlünk, olyanokat, amelyekhez az alkatrészeket — nem nagy pénzért — bárki meg tudja vásárolni. Kívánságát a HCC-ben dolgozó klubtagokhoz továbbítottam.

Agócs László, Borsodbóta

Bátorkodom szerény írásommal Önökhöz fordulni esetleges leközlés céljából. A PRIMO iskolai számítógép sok iskolában és magánzónál megtalálható, és pont az oly sokat felrótt dokumentációhiányt enyhítené írásom is.

Előkészületben van egy jól érthető, feladatokkal és mintaprogramokkal is ellátott PRIMO ALAPOK c. könyvem kézirata. Szintjét tekintve, eleinte szinte „szájbarágó" módon magyarázza el az alapokat, kezdeti fogásokat, majd középszintű ismereteken keresztül betekintést nyújtani a gépi kódolás alapjaiba is, mintaprogrammokkal. Függelékben egy bő kész programválaszték kerülne.

Kérem Önöket, hogy legyenek segítségemre abban, hogy ilyen kiadási ötletekkel hová lehet fordulni, valamint abban, hogy levelezéssel kapcsolatba léphessek haladó fokon programozó PRIMO-tulajdonosokkal, klubokkal.

Jó lenne, ha ügyelnének arra, hogy a lapjukban közölt programok gépbe írás után fussanak is. Tele vannak vagy sajtó- vagy logikai hibákkal. Azt fel sem merem tételezni, hogy valamelyik szakértője a lapnak először nem nézi meg gépen is a leközlésre szánt programokat! Ilyen előzetes ellenőrzéssel a hibák kiszűrhetők lennének, valamint a nyomdászok figyelmét is fel kellene hívni arra, hogy számítógépes programok kiszedésénél a .,:: stb. nem mellékes dolgok.

Pl. az 1986. júliusi szám 8. o.-án a VULKÁN p. 130-as sorában a képletből egy zá-

rójelpár kimaradt és így nem fut. Helyesen:
130 F=Y+A(SQR((Y−87) (Y−87) 2,2+(X−128) (X−1128)+1))
A NEXT X:NEXT Y helyett NEXT X,Y is megteszi. Az apróbetűs printerprogramok alig kiböngészhetők.

Ennek ellenére lapukat a legjobb magyar ilyen jellegű újságnak tartom, és remélem nincs harag!?

PS: Inspiráló a lap őszinte, kritikus hangvétele is!!

Ellenkezőleg, őszinte véleményt várunk olvasóinktól. A lapban megjelenő anyagokat többször is ellenőrizzük, elsősorban a rovatvezetők, az olvasószerkesztő, és természetesen a felelős szerkesztő is. Ennek ellenére nem sikerül hibátlan lapot készíteni; a helyreigazítást köszönjük.

Dőry Zsolt, Budapest,

Szentendrei u. 32. 1035

(1985 évben vásárolt) LSI ATSZ kiadásban megjelent, 1001 játék és a graphics BASIC C-64-en című könyv közül egy fordítóprogramot, mely a KOALA—PAINTER-el készített rajzot átfordítja, használhatóvá teszi a Gr. BASIC számára.

Többszöri kísérlettel sem sikerült a közölt programmal az átfordítást elfogadhatóan végrehajtani. A jelentkező hiányosság, pl. egy egyszerű rajznál, alakzatnál;

KOALA-val készített rajz	programmal átfordítás	Gr. BASIC-ból behívott rajz

Természetesen egy bonyolultabb (háttér-) rajznál a jelenség még bántóbb.

Kérésem, — ha módjában áll — legyen szíves közölni, hogy a könyvben közölt program;

— **hibás-e,** (ha erre mód van, talán célszerű lenne a MIKRO MAGAZIN-ban leközölni a hibás rész javítását, aminek a könyvtulajdonosok bizonyára örülnének)

— **nem hibás** (az átfordítással próbálkozom tovább).

A levelét elküldtem az LSI ATSz-nek, de közöljük is, hátha valaki ismeri a probléma megoldását.

Litauszky György, Békéscsaba,

Petőfi u. 14. 5600

Egy rövid vélemény, ami egyre inkább kikívánkozik belőlem: Természetesen vannak régi számítógépek, és vannak újak is. Az is természetes, hogy az újabbak sokkal többet tudnak, és már olcsóbbak is, mint akár két évvel ezelőtt. Nyilván aki most akar számítógépet vásárolni, azt a korszerűbb technikára kell rábeszélni.

Türhetetlen szerintem viszont az a tendencia, ami főleg a BIT—LET hasábjain figyelhető meg. Legutóbbi számukban pl. egy VC—20 cikk előtt jegyzik meg: „Közlésén gondolkodtunk is, hiszen ahogyan ma már fel sem merül a ZX—81-es anyagok közlése, igazából a VC—20-as anyagokért sem vagyunk oda."

Tudomásul kellene venni azt is, hogy Magyarországon még mindig drága dolog a számítástechnika, és aki VC—20-ast vett, annak az a számítógép, aki Primót, annak az, és aki ezelőtt 3 évvel vett egy ZX 81-et 15 000 forintért, az most nem fogja a szemétbe dobni, mert megjelent a (szerintem nem is olyan kiváló) C-16. Nekem pl. most a Spectrum+ a számítógép, mert még másal nem volt dolgom, és mert sokkal többet tud, mint az eddigi japán zsebszámológépem.

Tudom, hogy egy lap anyagának összeállítása a főszerkesztő belügye, és hogy ezt a kritikát nem jó helyre címeztem. Szeretném viszont, ha a Mikroszámítógép Magazin továbbra is megmaradna olyan sokoldalúnak, amilyen eddig is volt, és így ne csak a Commodore-hívők forgathassák, mint a fent említett BIT—LET-et.

Voltaképpen csak megerősíteni tudom véleményét, szerintem már csak azért sem szabad µM-nek egy gép, pl. a C-64 mellett voksolni, mert az iskolákban még ma is a legnagyobb példányszámban előforduló gép a HT és a Primo, magánhasználatban pedig a Sinclair-ek. De nagyon sok Vic 20 és C-16 is van, különösen az amatőrök körében nagyon kedvelt a Homelab, de nem nehéz megjósolni, hogy rövidesen lesznek Videoton gyártmányú TVC-k is, nem is kevés számban. Ha néha túltengenének lapunkban a Commodore-írások, annak az az oka, hogy szerzőiktől kaptuk a legszínvonalasabb anyagokat, így nyilvánvalóan ezeket közöltük.

Pataki Károly 4440 Wolfen,

Eisenbahnstr. 23. NDK

Otthon töltött évi szabadságom alatt hallottam az Önök folyóiratáról (sajnos látni nem láttam), mely felkeltette érdeklődésemet.

Mivel én főfoglalkozásban számítástechnikával foglalkozom itt az NDK-ban, s ez ideig kizárólag önképzésre vagyunk utalva (közepes minőségű szakirodalom- ill. tanfolyambázison), szeretném, ha az Önök folyóiratához folyamatosan hozzájuthatnék.

A µM külföldi árusítása még mindig nem megoldott. Néhányan budapesti ismerőseikkel fizettetik elő, akik aztán elküldik a megfelelő címre. Egyelőre ezt a módszert ajánlom. Új kiadónkkal, az ILV-vel ismételten megkíséreljük a külföldi terjesztést megoldani.

Tóth Viktor, Oroszlány,

Haraszthegyi út 1/b. 2840

Nagy érdeklődéssel olvasom lapjukat, és egy hóbortos ötletem miatt zavarom Önöket.

Szüleimtől egy C-64-est kaptam, és a barátommal elhatároztuk, hogy ezzel az ötlettel Önökhöz fordulunk. A kérés pedig nem lenne más, mint az, hogy milyen módon lehetne nagyobb anyagi ráfordítás nélkül egy C-64 és egy Spectrum gépet összekötni nagyobb távolságban, és esetleg milyen alkatrészek és anyagok kellenének hozzá.

Még egy kérésem lenne, az, hogy hol lehet Simon's BASIC bővítőt kapni.

Dr. Simonyi Endre válasza:

Egy számítógéppár összeköthető nagyobb távolságból telefonvonalon keresztül, ha

— a gépeknek van soros be- és kimenete,
— ha van modem,
— ha van telekommunikációs szoftver.

A kérdezett gépekhez ilyen van, de ennek az elkészítése meglehetősen sokba kerül.

A Simon's-BASIC ügyben a HCC tud segíteni, ha klubtag. A klub tagja csak NJSZT-tag lehet. A klubfoglalkozások szerdán 3—6 óra között vannak a BME F. épületében a II. épületszárnyban II. em. 9. sz. alatt.

Horváth Katalin, Budapest

Fém u. 4.

Levelemmel utószót szeretnék írni a µMagazin ez évi márciusi számában megjelent „Tankönyv vagy ifjúság elleni bűntett", valamint a júliusi számban olvasható „Válasz egy tankönyvbírálatra" című Pogány Csaba kontra Simonovits Miklós szópárbajra.

Annak ellenére, hogy matematika-fizika-számítástechnika tanár létemre szakmai szempontból is lenne véleményem, én most emberi oldalról szeretném megközelíteni a kérdést és néhány gondolatot hozzáfűzni az olvasottakhoz.

Meg kell mondanom, hogy én először a július: „védekezést" olvastam, és az annyira megdöbbentett, hogy előkerestem a márciusi cikket, hogy megtudjam, mi lehetett az a „támadás", amely ezt a határozott hangú, felháborodott, ugyanakkor kicsit elkeseredett választ kiváltotta. Megtaláltam, elolvastam, és fel vagyok háborodva. Mindenekelőtt azért, mert Pogány Csaba cikke megjelent.

Ebben a cikkben a szerző ifjúság elleni bűntettet emleget, didaktikáról, pedagógiáról, munkára nevelésről ír csupa-csupa elmaraszt alót. Ugyanakkor nem veszi észre, hogy írásával ő követi el a legnagyobb emberi, pedagógiai, nevelési hibát: diákjai előtt gúnyos, kötözködő, rosszindulatú, nagyképű stílusban marasztal el egy tanárt (nem beszélve a többiekről). Rendkívül etikátlan magatartásnak tartom.

Az már a tanár—diák viszonytól is független véleményem, hogy bármilyen könyv esetében emberhez méltó hangon, kulturáltan, segítő szándékkal is meg lehet fogalmazni egy kritikát. Különösen igaz ez akkor, ha egy ilyen, még úttörőnek számító vállalkozásról van szó, mint amilyen ez a tankönyv; egy ilyen tudományterületről, ahol még nincsenek kikristályosodott, esetleg megkövesedett didaktikai, módszertani elvek. A számítástechnika oktatásának ebben a kezdeti időszakában az érdekelteknek inkább össze kellene fogni, egymást segíteni, hogy minél hamarabb, minél jobb oktatási módszereket találjanak minden korosztálynak.

Köszönet dr. Urbán Jánosnak, hogy ha már így alakult, és a cikk — ami ezt az ügyet semmiképp sem vitte előbbre — megjelent, kérte a nyilvánvalóan maximálisan jószándékú, a gyerekeket szerető és a nyelvüket értő szerző válaszának közzétételét.

Tanulságos levél, ezért közöljük.

Kedves olvasóink! A szerkesztőség valamennyi munkatársa nevében kellemes ünnepeket, boldog és sikeres új évet kíván:

Kovács Győző

Zeneszerzés

Korunk varázsdoboza a számítógép, életünk egész területét behálóza. Így természetesen a művészetekét is.

A zene területén a zenetudomány, a népzenekutatás és a zenepedagógia sikerrel használja a számítógépeket. Kevesebben ismerik, hogy a számítógépnek fontos szerepe lehet a zeneszerzésben is, a gép új lehetőséget ad a zeneszerzőnek.

Először is felmerül a kérdés, miért van szükség a zeneszerzésben a számítógépre? Erre roppant egyszerű a válasz. Azért, mert a számítógép és elektroakusztika segítségével behatolhatunk a hangok mikrovilágába. A hangok mikroelemeinek és fizikai tulajdonságainak megváltoztatásával és egymásra hatásával (feszültségvezérlés, frekvenciamoduláció stb.) eddig még soha el nem képzelhető új hangzásokat és hangszíneket állíthatunk elő. Ennek a hangi mikrovilágnak az eredményeit az elektroakusztikai zene már alkalmazza, analóg elektronikai berendezésekkel új hangzásokat és hangszíneket tudnak előállítani. A számítógéptechnika alkalmazásával a sokszor órákat igénybe vevő hangátalakításokat percek alatt meg lehet valósítani. A hagyományos hangszereket és az emberi énekhangokat, vagy akármilyen konkrét hangot is tetszés szerint feldolgozhatunk, és ezáltal meghökkentő és bámulatos hangzásokat nyerhetünk. A digitális hangfeldolgozással a zene vertikális (harmoniai), horizontális (szólamszerűség és dallam) és formai megoldásainál is felbecsülhetetlen és gyors segítséget tud egy jól szerkesztett számítógépes zeneprogram adni a zeneszerzőnek.

Az információtechnika szédületes és páratlanul gyors — és bizony csak nehezen követhető — fejlődése beláthatatlan. Valószínűleg e technika művészi és zenei alkalmazásának még csak a kezdetén tartunk. Mindenesetre az az aggodalom, hogy a zene elmechanizálódik a számítógép által, kevéssé megalapozott, mivel a gép csak az emberi logika és művészi fantázia által megírt program megvalósítását végzi, még akkor is, ha ez a program logikai döntést kíván a géptől.

A világ számos elektroakusztikai stúdiójában rendkívül sok rendszer, nyelv, program, valamint gépi berendezés létezik a zenei hanganalízis/szintézis, feldolgozás és komponálás területén. Ezek közül egyik legjelentősebb úttörő munkát a BELL-Laboratórium (USA) végezte. Itt készítette Hiller és Isaacson a statisztikai analízis alapú számítógépes program segítségével megvalósított kompozícióját, az ILLIAC SUITE c. vonósnégyesét 1957-ben. G. M. Koenig az utrechti Sonologiai Intézetben, szeriális zene szintézisű művét, a Project I és Project II-t, PDP 15 gépen FORTRAN IV. programnyelv segítségével komponálta. Párizsban, Xenakis a stochasztika, az aleatoria, valamint a valószínűségszámítás elvei szerint vezérli számítógépes zenei programjait. A Bell-Laboratóriumban (USA) Max Mathews a digitális additív szintézisen alapuló MUSIC IV, később a MUSIC V. programnyelvet alakította ki. Ennek a nyelvnek a továbbfejlesztése a MUSIQUE X., amit a párizsi IRCAM-nál használtak. (Institute de Recherche et Coordination Acoustique/Musique)

Egy rövid cikk keretében nem lehet vállalkozni ennek a területnek az áttekintésére, ezért csak egy módszert szeretnék röviden egy példával ismertetni: a szintetikus zene előállítására szolgáló ADDITÍV SZINTÉZIS-t, ami végül is hangszalagra rögzített zenét eredményez.

Ez a hangszintézis úgy történik, hogy egy meghatározott programnyelven — jelen példánknál ez a nyelv a C. MUSIC — írt utasításokat egy terminálon keresztül adjuk meg a számítógépnek. Egy analóg-digitális konverter (ADC) a programot átalakítja digitális jelekké, és innen a számítógép központi memóriájába, illetve központi feldolgozóegységébe kerül a most már bináris program. Ebből az egységből a feldolgozás után az anyag átkerül egy újabb, most már digitál-analóg konverterbe (DAC), ami a bináris kódokat feszültségekké alakítja át. Ezek a feszültségek vezérlik a különböző hangkeltő egységgenerátorokat és egyéb készülékeket, amik a végső hangzást eredményezik.

A mintavételi frekvencia 45K Hz, ami azért mondható ideálisnak, mert a még hallható legmagasabb hangból is legalább két minta vételét lehetővé teszi, és így az egész hallótartományt átfogja és hűen visszaadja.

A bemutatott komponálási példa egy C. MUSIC nyelvvel készült program, a kompozíció (a partitúra elkészítése) három fő részből áll:

A) a „hangszerek" meghatározása (tehát a különféle egységgenerátorok (unit generátor) összekapcsolása.

B) Az egységgenerátorok által használt függvényeknek függvénytáblázatok alapján való megadása, amik a hangok burkológörbéit meghatározzák és a vezérlőfeszültségeket kialakítják.

C) Az ezeken a hangszereken megszólaló hangok meghatározása és összefűzése.

1984-ben a párizsi IRCAM-nál elkészítettem a Studium Digitale c. darabomat. Ennek a darabnak egy partitúrarészét, egy hangszer — *insfm* — és ennek függvénymeghatározásait és az ezen a „hangszeren" előállítandó hangokat jelző programot részletezem az alábbiakban. A darabban még másik öt „hangszer" is szerepel.

A „hangszer" megszólaltatásához tudnunk kell, hogy milyen hangforrást használhatunk. A „hangszerek"-et egységgenerátorokból lehet összeállítani. A C. MUSIC rendszerben számos egységgenerátort használunk, az *insfm* — „fantázia hangszer" az OSC, az OUT, az ADN, a MULT és a SEG generátorokból van összekapcsolva. Természetesen nemcsak „fantázia" hangszereket lehet ilymódon összeállítani, hanem bármely hagyományos hangszert is szimulálni lehet az igazi hangszer spektogram analízise alapján, pl. egy Stradivárit is, de egy tucathegedűt is. Az *insfm* hangszerben használt egységgenerátorok szerepe a következő:

1. Oszcillátor (OSC) *(1/a ábra)*

Az oszcillátornak két bemenete van, az egyik az amplitudót határozza meg (a bal oldali), a másik a frekvenciát. Az oszcillátorokban meg kell adni a függvénytáblázatot, ami tulajdonképpen a keletkező jel alakját határozza meg. A függvénytáblázatokat egy könyvtárból lehet lehívni, vagy a függvény pillanatnyi értékei megadásával tetszőleges táblázatokat lehet készíteni. A kimenetparaméter a következő egységgenerátorral való kapcsolatot rögzíti. A pillanatnyi értéktároló paraméterek szerepe, hogy a kiszámolt (generált) hang egyes diszkrét mintavételi értékeit a számítás végéig tárolja, és utána továbbítja a következő egységgenerátorhoz.

Az utasítás paraméterei:

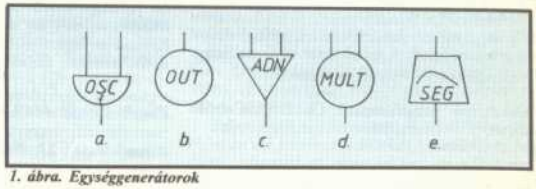
OSC kimenet amplitudó lépésköz (Fr) függvénytáblázat pillanatnyi értéktároló .

2. OUT Kimeneti egységgenerátor *(1/b. ábra)*

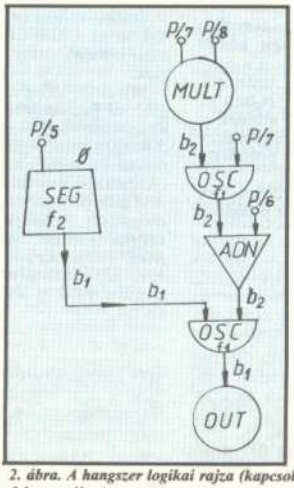
A kimeneti egységgenerátor fogadja a kész végső hangzás diszkrét mintavételi értékeit és tárolja. Egyetlen paramétere van, a bemenet, ami azt adja meg, hogy a hangzást melyik egységgenerátor továbbította.

Az utasítás paraméterei:

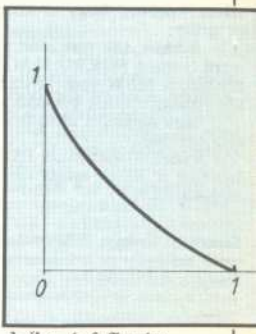
OUT bemenet



1. ábra. Egységgenerátorok



2. ábra. A hangszer logikai rajza (kapcsolás/folyamatábra)



3. ábra. Az f₂ függvény

3. ADN Összeadó *(1/c. ábra)*

Az összeadó egységgenerátor több hangjel összeadására szolgál. A bemenetparaméterek az előző egységgenerátorokkal való kapcsolatokat rögzítik.

Az utasítás paraméterei:

ADN kimenet bemenet bemenet

4. MULT Szorzó *(1/d. ábra)*

A MULT egységgenerátor pontosan úgy működik, mint az ADN, csak a bemeneti jeleket nem összeadja, hanem megszorozza. Ezt az egységgenerátort nagyon jól lehet használni a modulációtechnikánál.

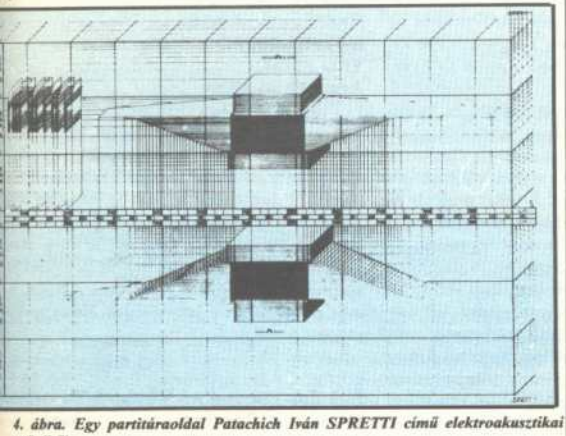
Az utasítás paraméterei:

MULT kimenet bemenet bemenet

5. SEG Szegmens egységgenerátor *(1/c. ábra)*

számítógéppel

Patachich Iván a párizsi Pompidou Center számítógépének terminálján



4. ábra. Egy partitúraoldal Patachich Iván SPRETTI című elektroakusztikai művéből

Ez az egységgenerátor a meghatározott ideig tartó hangamplitudó burkológörbéjét határozza meg. A kimenetparaméter határozza meg a következő egységgenerátorral a kapcsolatot. A bemenetre kerül az amplitudóérték. A függvénytáblázattal lehet megadni a burkológörbe alakját. Itt is szükség van a már ismertetett pillanatnyi értéktárolásra is. A lépésközparaméter adja meg a végső hang burkoló görbéjét (az OUT-ban), azaz, hogy a hang milyen hosszú ideig (a teljes időtartam hányad részéig) fog szólni. Ha ez a paraméter Ø, akkor a teljes időtartam alatt végig szól.

Az utasítás paraméterei:

SEG kimenet amplitudó függvénytáblázat pillanatnyi értéktárolás lépésköz

Ezekből az egységgenerátorokból elkészíthetjük a „hangszerünket" a jelen esetben az *insfm*-et. E hangszer egy frekvencia-modulált hangzást állít elő — innen ered az insfm akronim elnevezés. — A burkológörbéjét egy szegmensgenerátor vezérli. A P/6 vivőfrekvenciát egy P/7 értékű frekvencia modulálja, b_2 mértékben, egy összeadón (ADN) keresztül.

Ezután elkészíthetjük e „hangszer" logikai rajzát (kapcsolás vagy folyamatábráját). *(2. ábra)*

Az insfm hangszer programja:

INS Ø insfm: Ez az első utasítás azt jelenti, hogy az *insfm* hangszer a Ø. időpillanatban elkezd játszani. Ezután leírjuk a 2. ábra szerinti logikai kapcsolást utasításokkal. A programban használt paraméterek a következőket jelentik:

SEG b_1 P/5 f_2 d Ø; P/5 amplitudó, a hangerő dB-ben
MULT b_2 P/7 P/8; P/6 a vivőfrekvencia
OSC b_2 b_2 P/7 f_1 d; P/7 a moduláló frekvencia
ADN b_2 b_2 P/6; P/8 a frekvenciamoduláció mértéke
OSC b_1 b_1 b_2 f_1 d;
OUT b_1;
END;

A hangszer programjába ezután be kell írni a hangok előállításához használt két függvénygenerátort.

Az insfm hangszer függvénytáblázatait előállító függvénygenerátorok (f_1 és f_2)

Az f_1 generátor sinus függvényt állít elő (GEN5), a függvény előállítására szolgáló utasítás a következő: GENØ GEN5 f_1 1 1 Ø;
Ezen utasítássor jelentése Ø időnél induló GEN5 (sinus) típusú, f_1 nevű, 1 felhangot, 1-es maximális amplitudót előállító, Ø-fázisú függvény.
A következő f_2 generátor exponenciális függvényt (GEN4) állít elő. *(3. ábra)*
GEN Ø GEN4 f_2 Ø, 1, −5, 1, Ø;

A jelentése Ø időnél induló GEN4 (exponenciális) típusú f_2 nevű függvény, ami a Ø időnél 1-es amplitudóval indul, majd 5-ös meredekséggel exponenciálisan csökken, és végül eljut az 1 másodpercnél Ø-ra.

„Elkészítettük" a hangszert, ezután már csak meg kell szólaltatni, t.i. ezen a hangszeren már le lehet „játszani" különböző zenei struktúrákat, ha megírjuk a struktúrák programját. A feladat, hogy a BASS 7/d nevű, zenei struktúra programját elkészítsük.

A strukturát hangonként (NOTE) kell „lekottázni," a hangokat paraméterekkel kell megadni. Egy hangutasítás szerkezete a következő:
P/1, P/2, P/3, P/4, P/5, P/6, P/7, P/8.

A programban használt további paraméterek:

P/1 — NOTE — azt jelenti, hogy egy zenei hang megszólal.
P/2 — a kezdő idő (belépési idő).
P/3 — a hangszer neve, példánkban *insfm*.
P/4 — a hang időtartama.
P/5 — amplitudó(hangerő).
P/6 — vivőfrekvencia.
P/7 — moduláló frekvencia.
P/8 — a fekvenciamoduláció mérete.

Ezek után a BASS 7/d struktúra programja:

BASS 7/d.
NOTE Ø insfm .5 — 9dB 87Hz 90Hz 3;
NOTE .5 insfm .25 — 9dB 131Hz 135 Hz 3;
NOTE .75 insfm .25 — 9dB 87Hz 90Hz 3;
NOTE 1. insfm .25 — 9dB 131Hz 135Hz 3;
NOTE 1.25 insfm .5 — 9dB 87Hz 90Hz 3;
NOTE 1.35 insfm .25 — 9dB 131Hz 135Hz 3;
NOTE 2. insfm .5 — 9dB 87Hz 90Hz 3;
NOTE 2.5 insfm .25 — 9dB 131Hz 135Hz 3;
NOTE 2.75 insfm .25 — 9dB 87Hz 90Hz 3;
NOTE 3. insfm .5 — 9dB 131Hz 135Hz 3;
TER;

Most már csak azt kell felismerni, hogy a BASS 7/d zenei struktúra egy basszus szólamú 7/8-as ütemű bolgár ritmusú dallam.

Az insfm-hangszeren lejátszott BASS 7/d nevű struktúrát ezután már egy végső keverő programba bármilyen változtatással vagy átalakítással (transzponálás, augmentálás, diminuálás, gyorsítás, lassítás stb.) be lehet szerkeszteni.

A párizsi IRGAM-nál a leírtnál lényegesen fejlettebb rendszerrel dolgoznak a zeneszerzők. Az ott működő 4 X rendszerbe be lehet vinni a hagyományos akusztikus hangszerek hangját, vagy akár az énekhangot is, ezt a gép a zeneszerző programjának megfelelően néhány ezredmásodperc alatt feldolgozza, és a hangszeres zenével, illetve az énekkel együtt (real time módon) megszólaltatja. Ily módon a számítógép az „élő hangversenyeken" is szerepet kap, mert ezeket a hangátalakítást eredményező számításokat a helyszínen nagy sebességgel végzi el, és az eredmény, az átalakított hangzásélmény az élő zenével egyszerre hallható.

Az elektronikus — számítógépes zenét komponáló zeneszerző néha kénytelen a zenei struktúrát nem a megszokott módon lekottázni.

A *4. ábrán* egy elektroakusztikai partitúra látható, amelyet le lehet játszani pl. analóg elektronikus eszközökkel is, vagy el lehet készíteni ennek a komplex hangstruktúrának a számítógépes programját is. Az ábra egy kétcsatornás sztereofonikus művet mutat be, a hangzást két egymás fölé helyezett háromdimenziós koordináta-rendszerben ábrázolja (az X tengelyen az időt jelezzük sec-ben, illetve magnetofonszalag cm-ben, — 38-as sebesség —, a logaritmikus osztású Y tengely 10-től 10 000-ig a hangmagasságot ábrázolja, a Z tengely az amplitudót 0-tól 50 dB-ig.)

A palackból kieresztett jó „hangszellem" remélhetőleg még sok csodát tartogat számunkra, felfedve előttünk a mikroakusztika kevéssé ismert rejtelmes vidékét.

Számítástechnikai statisztikai évkönyv, 1985.
(Budapest, 1986. Statisztikai Kiadó Vállalat.
Ára: 70,— Ft.)

A kötet a korábban hasonló címmel megjelent kiadványok szerves folytatása; biztosítja a publikált adatok továbbvezetését, és ezáltal módot nyújt az összehasonlításra.

Az évkönyv számot ad számítástechnikai eszközállományunk mennyiségi és értékbeni fejlődéséről, műszaki színvonaláról, összetételéről és kihasználtságáról. Ismerteti a számítógépen végzett munkák és számítástechnikai alkalmazások jellegét, mennyiségét, bevételeit és költségeit, valamint a szakterületen foglalkoztatottak létszám-, szakképzettségi és béradatait.

A mutatók alapvető népgazdasági ági, ágazati, SZAB felügyeleti és szektoronkénti csoportosításban szerepelnek. A nemzetközi összehasonlítást néhány összefoglaló, a világ, az USA és az európai tőkés országok eredményeit bemutató tábla szolgálja.

Erdős Iván:
IBM PC/XT információs kártya
(Budapest, 1986. LSI ATSZ, 72 oldal.
Ára: 135,— Ft.)

A jól kezelhető segédkönyv első fejezete az IBM PC/XT-vel kapcsolatos technikai információkat — karakterkódok, memóriafelosztás stb. — tartalmazza. Az MS—DOS referencia részben a parancsok formátumát, az Edlin parancsokat ismerteti, BASIC és Intel 8086 programozási segédletet közöl: parancsokat, utasításokat, hibaüzeneteket.

Bodó László — Urbán Gábor
Zsebszámológép-programok építőmérnököknek
(Budapest, 1986. Műszaki Könyvkiadó, 163 oldal.
Ára: 58,—Ft.)

Az utóbbi években az országban ugrásszerűen megnőtt a programozható kalkulátorok népszerűsége. Ellentétben a nagyszámítógépek használatával, amelyhez típusonkénti kimerítő ismertetésre van szükség, a kalkulátorok esetében a kezelésben mutatkozó analógiák miatt elegendő egy ún. gépcsalád-programkönyvtár létrehozása. Ebben egyetlen könyv tartalmazza a gépcsaládra vonatkozó általános programozási ismereteket.

A könyv anyaga három fő részre tagolódik: 1. Programozható zsebszámológépek programozásának technikája. 2. Programgyűjtemény néhány konkrét számológépre, a géptípusok műszaki adatainak ismertetésével. 3. Algoritmus-, illetve programozandó feladatgyűjtemény, főként az építőipar területéről.

Bóna Gábor — Erényi István — Vajda Ferenc:
Több-mikroprocesszoros rendszerek
(Budapest, 1986. Műszaki Könyvkiadó, 313 oldal.
Ára: 93,— Ft.)

A szerzők célja az volt, hogy a több-mikroprocesszoros rendszerek hardverjének és szoftverjének felépítését, a legfontosabb architektúrák jellemzőit bemutassák az olvasónak. Foglalkoznak az ilyen rendszerek használatához, programozásához és kidolgozásához szükséges alapelvekkel és módszerekkel. A könyv olvasása feltételez bizonyos számítástechnikai, mikroprocesszoros technikai alapismeretet. A kötet nem tér ki a különböző alkalmazások ismertetésére, a hangsúlyt az alkalmazásoktól független, általános, minden rendszerben valamilyen formában megtalálható felépítési elvek, kidolgozási eljárások bemutatására helyezi. A főbb fejezetek: A több-mikroprocesszoros rendszerek áttekintése, architektúrája; Sínrendszerek; Párhuzamos mikroprocesszorok alkalmazásának matematikai és szoftver eszközei; Különleges LSI/VLSI elemek és alkalmazásuk.

Hornig — Trapp — Weltner:
Tippek és trükkök a Commodore 64-eshez
(Budapest, 1986. Data Becker — Novotrade, 187 oldal.
Ára: 239,— Ft.)

A sokat ígérő alcím szerint a kötet a C64-es felhasználók kincsesbányája. Valóban, az olvasó számos ötletet és trükköt talál házi használatra, a kazettás egység vezérlésének lehetőségeitől a különböző BASIC-tippekig. Külön fejezet ismerteti a szoftvervédelem módjait, a sajátkezű parancsbővítést, a grafikai lehetőségeket, a játékok programozásának néhány ötletét. A függelék igen hasznos táblázatokat tartalmaz.

Commodore Plus/4.
A beépített programok kezelése
(Budapest, 1986. Novotrade, 154 oldal.
Ára: 99,— Ft.)

A könyv a Plus/4-be beépített négy programmal foglalkozik: a szövegszerkesztő, a számviteli tömb, a grafika és az adatbázis programokat ismerteti számos gyakorlati példán. Kiemeli a programok könnyű kezelhetőségét, azt az előnyüket, hogy a felhasználó az egyes programok információit egy gombnyomással át tudja a másik programba vinni.

Commodore Plus/4 felhasználói kézikönyv
(Budapest, 1986. Novotrade, 160 oldal.
Ára: 99,— Ft.)

A kötet a valóban otthoni alkalmazásra szánt Commodore Plus/4 személyi számítógép használatáról szól. Részletesen ismerteti a gép összeállítását és üzembe helyezését, a billentyűzetről elvégezhető különböző feladatokat, megtanít a különféle szoftverek használatára; önálló fejezeteket szentel a matematikai, a grafikus, a hang és más programozási lehetőségek kihasználásának.

Pál Zsuzsanna — Révbíró Tamás
Hetedhét Commodore — Plus/4
(Budapest, 1986. Novotrade, 145 oldal.
Ára: 79,— Ft.)

A Hetedhét sorozat újabb darabja ismét egy Commodore számítógéppel foglalkozik. A korábbi, Commodore 16-ról szólóval szinte szóról szóra azonos a kötet, mivel a két gép fizikai felépítésétől eltekintve majdnem azonos. A különbség a Plus/4 négyszer nagyobb szabad memóriájában és a gyárilag beépített felhasználói szoftverekben van.

A könyv célja, hogy a kezdőket megbarátkoztassa a Commodore BASIC-kel, korunk egyik legelterjedtebb számítógépes nyelvével, és hogy megismertesse magát a gépet.

0,8—1,6 Mbájtos tároló

Félmagas, 8 hüvelykes, kétszeres sűrűségű hajlékonylemezes tároló gyártását kezdték meg a Magyar Optikai Művekben. A félmagas kivitel lehetővé teszi, hogy a gépekben a hajlékonylemezes tárolók számára kihagyott szabványos rekeszekbe egy helyett két tárolót építsenek be. A tárolási kapacitás formattáltan is 0,5 vagy 1 Mbájt, az egyszeres vagy kétszeres írássűrűség alkalmazásától függően. Az MF 6400 DSL jelzésű tároló mechanikai méretei és műszaki paraméterei alapján kompatibilis a Shugart cég hasonló tárolójával.

Az Iszkra-család

A leningrádi Lenelektronmas egyesülés kifejlesztett egy új professzionális, az IBM PC-vel kompatibilis mikroszámítógép-családot, az Iszkra 1000 sorozatot. Az alapmodell az Iszkra 1130, mely az Intel 8086 mikroprocesszornak a Szovjetunióban gyártott KR1810VM86 típusjelzésű változatát használja. Operatív tára 256 kbájtos. Monochrom a megjelenítője, és két, K 5600.20 típusú, 360 kbájt kapacitású hajlékonylemezes tárolója van. Ára Iszkra 041SM típusú nyomtatóval 11 500, Robotron gyártmányú, K 6312M típusú nyomtatóval pedig 12 000 rubel.

Az Iszkra 1030 magából az Iszkra 1130 típusú alapmodellből és egy hozzá csatlakoztatott bővítőből áll, amely méretben megegyezik az alapgéppel. Ez lehetővé teszi a következő perifériák csatlakoztatását: bolgár gyártmányú SZM 5400 vagy SZM 5408 típusú, nagy átmérőjű merevlemezes tár, sorrendben 5 vagy 6 Mbájt kapacitással; szintén bolgár, SZM 5300.01 típusú félhüvelykes mágnesszalagos tár.

Az Iszkra 1030 tizenkét adatátviteli csatornát is tartalmaz, s a nagyobb terhelhetőség érdekében operatív tára 1 Mbájtra bővíthető. Az Iszkra 1031 felépítése — a merevlemezt és a mágnesszalagos tárat kivéve — megegyezik az Iszkra 1030-éval, de az előbbiek helyett van két másik perifériája: egy N307 típusú rajzgép és egy digitalizáló. A kiépítésből látható, hogy ezt a tervezői munkák automatizálásának támogatására tervezték.

A család az ADOS operációs rendszer vezérlése alatt működik, amely az MS—DOS funkcionális megfelelője. A magas szintű programnyelvek közül egyelőre csak BASIC nyelven programozható, s van hozzá MASM néven makroassembler fordító is.

Az Iszkra-család érdekessége, hogy hardver szempontjából nem kompatibilis a Minszkben kifejlesztett ESZ 1840-nel. Különbözőek a panel- és a blokkméretek, a perifériák, bizonyos mikroáramkörök, különböző a billentyűzet elhelyezése, az információkódolás stb. Ez is utal arra, hogy egymástól függetlenül végezték a fejlesztést. A jövőben várható több párhuzamosan kialakított típus megjelenése, mert egész sor szovjet minisztérium irányítása alatt folyik már mikroszámítógépek tervezése és gyártásba vétele.

IBM PC magyarul

Hazánkban már csaknem kétezer IBM PC és azzal kompatibilis mikroszámítógép működik, így már tömegesnek nevezhető az igény, hogy a számítógéppel készíthető leveleket, szövegeket ezen a típuson is a magyar helyesírás szabályai szerint végezhessük. A SZÁMALK új programtermése lehetővé teszi a magyar betűkészlet használatát, nemcsak a képernyőn, hanem a mátrixnyomtatón is. Az utóbbi esetben kétféle betűkép és nyomtatási sebesség is rendelkezésre áll a tisztázat és a piszkozat készítéséhez. A nyomtatón többféle betűtípus használata is biztosított. Fontos előnye ennek a programtermékeknek, hogy változtatás nélkül együttműködik a legtöbb, általában használt programmal. A rugalmas alkalmazást az is biztosítja, hogy a terminál billentyűzete egyetlen gombnyomással átkapcsolható magyarról angol (német) ábécés üzemmódba és fordítva.

Paprikamolnárok

A Szegedi Élelmiszeripari Főiskola munkatársaiból alakult munkacsoport — dr. Huszka Tibor vezetésével — az utóbbi hónapokban olyan rendszert fejlesztett ki, amely nagy biztonságot garantál a paprikaőrlemény kívánt minőségének elérésére, s közben még a feldolgozási költségeket is csökkenti. A mikroszámítógép a paprikafeldolgozóban lévő összes nyersanyag mozgását követi.

A rendszer alkalmazása esetén a gyár két hétre előre el tudja dönteni, hogy édesnemes, rózsa-, csemege- vagy csípős paprikát készítsenek-e, gondoskodik az adott változat gyártásához szükséges tárolók feltöltéséről, a gyártás során pedig igyekszik a technológiának megfelelően a lehető legrövidebb időn belül minden maradékot kiüríteni a tárolókból.

Az új, számítógépes eljárás a legtapasztaltabb paprikamolnárok keze alól kikerülő őrlemény minőségét garantálja, sőt a szín pirosságát illetően felül is múlja azt.

A Szegedi Paprikafeldolgozó Vállalat szőregi telephelyén a fűszerpaprika számítógéppel optimalizált feldolgozásának modellje már működik, az iparszerű bevezetés várható időpontja egyelőre nem ismeretes.

Optikai olvasás

A Workless Station olyan íratolvasó berendezést fejlesztett ki, amely írógéppel írt szöveget mikroszámítógépbe olvas, ezáltal megkíméli felhasználóját a már előzőleg leírt szöveg újra leírásától. A berendezés nyolc különböző betűtípust ismer, de ez szoftver úton tovább bővíthető. Olvasási sebessége percenként 4 db A/4 méretű oldal. Az olvasó a szabványos RS232 illesztővel szinkron vagy aszinkron módon csatlakoztatható perifériaként a mikroszámítógéphez.

Számítógépes gyártásirányítás

Megkezdődött a termelés egészének számítógépes irányítása a győri Rába Gyárban. A nagyüzem minden részlegében különféle teljesítményű számítógépeket helyeztek el, s ezeket hálózatba kötötték. A terminálok száma 350, a nyomtatóké 100. A munka során 150 kilométernyi kábelt fektettek le. A postától külön telefonvonalakat béreltek a győri központ és a vidéki telepek számítógépes kapcsolatához.

A Rába oktatási központjában a számítógépes képzést már a múlt évben megkezdték, s azóta is folyamatosan végzik. Több mint háromezer embert kellett felkészíteni az új feladatra: a gyár 17 500 dolgozója közül ennyien kerültek közvetlen kapcsolatba a számítógépekkel.

A számítógépes irányításra való áttéréssel javul a készletgazdálkodás, a munka szervezettsége. Csökken az az idő, ami a piaci igények felmérésétől az újabb konstrukciók gyártásáig eltelik, így a főleg exportra termelő gyár gyorsabban igazodhat a kereslethez. Nem közömbös persze az a nyereség sem, amit az adminisztrációban könyvelhetnek el.

Commodore nyomtató Spectrumhoz

Igen sok intézmény használ hazánkban Commodore rendszereket, s közülük soknak van Sinclair Spectrum számítógépe is. Ezek számára készített olyan illesztőt a Háziszámítógép-építők Klubja, a HCC, amely lehetővé teszi az elterjedt Commodore nyomtatóknak (MPS—801, MPS—802, MPS—803, Seikosha GP100, VIC—1526 stb.) a Sinclair Spectrum géphez való csatlakoztatását. Karakterkészlete

40, 80, 120 jel/sor megjelenítésére alkalmas. Beépített ékezetes és egyéb rögzített felhasználói karakterei miatt használata kényelmes. Az illesztőn lévő NMI gomb benyomásával az aktuális képernyőről másolatot készít, majd a program futása folytatódik. A Sinclair Spectrum operatív tárából helyet nem igényel. Működik gyári programokkal, szövegszerkesztőhöz is használható, prospektusok, nyomtatványok, illusztrációk készítéséhez jól hasznosítható. A vállalati alkalmazásokon kívül az iskolai, kultúrházi, reklámirodai felhasználása lehet még gyakori. 6900 forintos ára ahhoz képest alacsony, hogy ennyiért a felhasználó nyomtatót kap a Commodore-ok mellett meglévő Spectrum gépéhez is.

0,25 μm

A japán Toshiba cég elkészítette egy 0,25 μm mintázatfinomságú áramkör prototípusát. Ezt a sziliciumszelet mintázatkialakítás előtti gyenge, homogén elektronsugaras besugárzásával érték el. Az új eljárás alkalmas lesz a 4, 16 sőt a 64 Mbites tárolómorzsák gyártására is.

Szovjet profi gépcsalád

Még a múlt év decemberében sikerrel folyt le a minszki számítógépgyárban az ott készített ESZ 1840 típusú új szovjet, az IBM PC-vel kompatibilis professzionális mikroszámítógép állami bevizsgálása.

Az ESZ 1840-be a szovjet gyártmányú K1810VM86 jelzésű mikroprocesszort építik, mely az Intel 8086 funkcionális megfelelője. Meghajtása 5 MHz, a regiszter—regiszter művelet sebessége 1,2 millió művelet/mp. Magának a gépnek a sebessége az IBM PC-nél 30%-kal nagyobb. A maximális operatív tár 1 Mbájt, az alapmodelleket 256 és 512 kbájtos tárral készítik. A gépet négy verzióban gyártják, melyek ajánlati árait 1987-re a táblázatban közöljük.

Az ESZ 1840-ből az idén 500-at, jövőre pedig már 10 ezret kívánnak gyártani. Időközben körvonalazták a fejlesztéseket is. Várhatóan 1986 végén készül el a Winchester-tárat is kezelő, az IBM PC XT-vel kompatibilis változat, ESZ 1841 típusszámmal. 1988 végére ütemezték az ESZ 1850 típusú gép elkészültét. Ennek operatív tára már 2 Mbájtig bővíthető, s képes lesz értelmezni a nagy ESZR gépek processzorutasításait is. A távlati tervekben szerepel a hálózati illesztők, a grafikus perifériák, a szakmai jellegű speciális perifériák és koprocesszorok kifejlesztése.

Az ESZ 1840 szoftverellátottsága az IBM PC-hez hasonló, de különbség a cirill betűk, továbbá a Szovejtunió népei speciális karaktereinek használati lehetősége. A gép az M86 operációs rendszer vezérlése alatt működik, amely a CP/M—86 funkcionális megfelelője. Egyelőre csak a BASIC interpreter áll a felhasználók rendelkezésére. Az Abak nevű program táblázatkezelési lehetőséget biztosít, a Telekszt pedig az ESZR gépek intelligens terminaljaként való működésre teszi alkalmassá az ESZ 1840-et. A gép jelentős alkalmazási korlátja a szövegfeldolgozási, adatbázis-kezelési, grafikai programcsomagok, valamint a korszerű programozási nyelvek (Pascal, C) fordítóinak hiánya, de kidolgozásuk folyamatban van.

Verzió	Operatív tár (kbájt)	A hajlékonylemezes tárolók száma	típusa	Ajánlati ár 1987-ben (rubel)
ESZ 1840	512	2	FD 55F	8610
ESZ 1840.01	256	2	FD 55F	7540
ESZ 1840.02	512	2	ESZ 5324	12050
ESZ 1840.03	256	2	ESZ 5324	10990

Átnéztem öt—tíz folyóiratot és kiválogattam az engem érdeklő híreket, cikkeket, mert úgy gondoltam, hogy ezek másokat is érdekelhetnek. Persze hogy kinek mi az érdekes, az nagymértékben függ a válogató személyiségétől, ismereteitől, szakterületétől. Lesznek olvasók, akik kevésnek, lesznek, akik érdektelennek, és lesznek, akik érdekesnek tartják beszámolómat. Ez végül is nem baj, hiszen sokfélék vagyunk — így kerek a világ.

Célom nem a szakirodalom áttekintése — erre témafigyelő szolgálatok léteznek —, csak a figyelem felkeltése. A forrásokat mindig pontosan közlöm, hogy akit a téma részletesebben érdekel, utánanézhessen. Nem szándékom a mikrogépes szakma minden területéről válogatni, és biztos, hogy nem mindenki számára lesz újság az újság. De valószínű, hogy ha száz folyóiratból válogatnék, akkor sem tudnék mindenkinek újat mondani.

Nézzük tehát, hogy a kiválasztott folyóiratok ez évi első 3-4 számából mi volt szerintem a legérdekesebb.

PC—WELT

(1986/1., 1986/2. Netzwerke für die PC—Host—Kopplung)

A cikk mintegy száz szoftvertermék piaci összehasonlítását közli. Áttekinti a PC — nagyszámítógépes hálózati termékeket, és nyolc jellemző alapján értékeli azokat. Egy-egy gyártó több termékével is szerepel. A számba vehető mikrogépek jelentős hányada IBM PC XT, AT, és a nagyszámítógépek jelentős része is IBM gyártmány.

A szoftvercsomagok szolgáltatásai igen széles skálán mozognak. Szerepel egyszerű terminálemuláció, fájlátvitel, de van SNA/SDLC és IBM token-ring (vezérelt jelzéses) protokoll is, 4 Mbit/s sebességgel. Ugyanilyen változatosak az árak is: 200-tól 20 ezer DM-ig terjednek. A termékek kb. 30 százalékának nem tüntetik fel az árát, ezek *auf Anfrage* (kérésre) megjelöléssel szerepelnek.

Az előállítók között természetesen első helyen a nagy számítógépgyártók — IBM, Borroughs, Siemens — szerepelnek. A mini- és mikrogépeket gyártó és fejlesztő kisebb cégek, valamint a csak szoftver előállításával foglalkozók teszik ki a lista másik felét.

CHIP

(1986/7. Kinz, R.: CHIP—TEST: Der neue Commodore 64)

A C64-esekből csak 1984-ben jó egymillió darabot adtak el. A sikert most javított burkolattal és korszerű operációs rendszerrel fokozza a Commodore. Az új C64 a régivel teljesen kompatibilis, ugyanakkor GEOS (Graphic Environment Operating System) operációs rendszere révén grafikus lehetőségei jócskán megnövekedtek. A gép processzora 6510-es. 64 kbájt RAM és 16 kbájt ROM tárakkal. Interfészáramkörei változatlanok: egy párhuzamos, egy soros (nyomtató, floppy stb.), két botkormány-, egy kazetta-, egy bővítőcsatlakozója van. Továbbra is monitor vagy tévé csatlakoztatható hozzá. A kompatibilitást a BASIC 2.0 változatával éri el.

A GEOS grafikus képességét látványosan az operációs rendszer „ablaka" jelenti.

Érdekes még a Geowrite nevű szövegfeldolgozó program, mely az értékelő szerint csak drágább számítógépeknél van.

A gép árát még nem közlik, csupán annyit, hogy az Egyesült Államokban a GEOS ára 60 dollár körül jár; így valószínű, hogy elődjénél drágább, de bátyjánál, a C128-asnál mindenképpen olcsóbb lesz.

(1986/8. Speichererweiterung für Commodore 16)

A C16-os gépek több szempontból érdekesek számunkra. Azonkívül, hogy számuk itthon is egyre növekszik, az általános iskolák számára ajánlott gép; nemcsak Nyugat-Európában, hanem nálunk is viszonylag olcsó — emlékezzünk a tavaly karácsonyi 7000 forint körüli akcióra. A C16 további előnye, hogy a BASIC 3.5 ver-

Azt írja az újság...

zió jól használható, és bővek a grafikus funkciói.

A cikk ismerteti a RAM 16-ról 64 k-ra való bővítését, belső RAM chip cseréjével. Tekintettel arra, hogy a gép NYÁK-felépítése igen egyszerű, a csere viszonylag könnyen elvégezhető. A C16-ban két 4 bit szélességű, 16 kbites RAM chip van (TMS 4416—20 vagy azonos felépítésű). Ezt a kettőt kell kicserélni TMS 4464—15 típusúra vagy azonos felépítésűre. Az utolsó két számjegy a hozzáférési idő ns-ban; 200-nak vagy annál kisebbnek kell lennie. A cserén túl két átkötést kell még megváltoztatni, hogy a RAM-ok 64 k-ig címezhetők legyenek.

A cikk lépésről lépésre ismerteti az átalakítást, figyelmeztet, mire kell vigyáznunk. Az átépítés után a C16 Plus/4-ként viselkedik majd.

Az ismertető külső segítséget is ajánl: egy kis céget, amely elvégzi az átalakítást. Hasonló munkára ajánlkozó hirdetését már a Mikromagazinban is olvastam.

IEEE SPECTRUM

(1986. június. Voelcker, J.— Wallich, P.: How Disks Are „Padlocked")

A szoftveripar évek óta műszaki, jogi és etikai kérdésekkel kínlódik, hogy megakadályozza a termékek illetéktelen másolását. Ez a tevékenység ugyanis egyidős a számítástechnikával. A helyzet a személyi számítógépek megjelenésével csak súlyosbodott, hiszen a forgalmazott termékek száma ez időre nagyságrendekkel megnőtt.

Programot tartalmazó floppyt tulajdonképpen kell is másolni, hiszen ha nincs biztonsági másolat, esetleg heteket kell várnunk, amíg az eredeti helyett az előállítótól újat kapunk.

Sok gyártó nem is védi termékeit, mert az „illetéktelen másolók" igen találékonyak. Akik viszont védelmet alkalmaznak, úgy gondolják, hogy legalább megnehezítik a másolók dolgát. A védelem egyszerű: készítsünk olyan programlemezt, amelyet egy közönséges személyi számítógép meghajtója el tud olvasni, de képtelen ugyanúgy megírni. Közönséges lemezen az adatokat koncentrikus körök mentén, sávokon rögzítik. A sávokat szektorokra osztjuk. Minden szektornak fejléce van, a felírás végén a hibák detektálására ellenőrzőösszeg található.

A legegyszerűbb védelmi eszköz, hogy egy vagy több „rossz" (ellenőrzőösszeg-hibás) szektort írunk a floppyra. Ha a védelmi program az adott helyen hibás szektort talál, a floppyn levő programot engedi betölteni és futni, egyébként — helyesen megírt ellenőrzőösszegű szektorok esetén — a programot törli. A védelmet az ún. bitmásoló programok hatástalanítják. Ennek ellenére ezt az egyszerű védelmet még néhány újabb gépnél, például az Atari ST-nál is alkalmazzák.

Extra, a szabványos mennyiségen felüli sáv(ok) beiktatása a védett lemezen megakadályozza a szokásos másolóprogramok használatát. Az eljárást a szabványosnál több sávot is másolni tudó másolóprogram hatástalanítja.

Az előzőhöz hasonlóan a sávon belüli szektorok számát a szabványos 8-9-ről, főként a külső sávokon, 10-11-re növelhetjük. Ezekbe a többletszektorokba lényeges programinformációk helyezhetők el. Hiányuk esetén a program nem futtatható, vagy törlődik. Természetesen extra szektort is másolni tudó másolóprogram ezt a védelmet is kijátssza.

A keménylemezek megjelenése a floppyn szállított programok másolását tette szükségessé, mivel ha másért nem, nagyobb sebességéért célszerű a keménylemezt használni. A keménylemezre másolt program csak ún. kulcsfloppy egyidejű használatakor fut. Az eljárást még különféle rejtjelezett fájlok, rekordok írása bonyolíthatja. Ezek a fájlok arról is felvilágosítást adhatnak, hogy az adott programot egyszer már üzembe helyezték. Ilyenkor a másodszori vagy többszöri üzembe helyezési kísérlet még az ún. kulcsfloppy alkalmazásával is sikertelen.

Az előbbieken kívül a forgalmazók a sávok és szektorok, az író-olvasó mechanizmus különféle megváltoztatásával igyekeznek a védett, illetve védeni kívánt termék egyediségét megőrizni.

A védelmi eljárások száma nő, a jogi vita erősödik, a cikk tanúsága szerint azonban nem sok eredménnyel. A felhasználók panasza a termékvédőkkel szemben az, hogy általában egyetlen készen vásárolt termék sem használható változtatás nélkül. Még jó, ha nincs benne hiba. Ha nem másolható, nem lehet javítani sem. Sokan a védelmet alkalmazók bojkottján gondolkoznak.

Terjed az a gyakorlat, hogy egy felhasználót több másolat készítésére jogosítanak fel, hiszen egy szervezet több PC-t is használhat. Az ilyen megoldást a több tíz, esetleg több száz munkahelyes helyi hálózatok még inkább indokolják, bár a szállítók idegenkednek ettől az eljárástól.

DATAMATION

(1986. június 1. RISC: Reality vs. Rhetoric. Moad, J.: Gambling on RISC. Bell, C. G.: RISC: Back to the Future? Serlin, O.: MIPS, Dhrystones and Other Tales)

RISC = Reduced Instruction Set Computers. A nagyhírű és -múltú folyóirat külön részt szentel ezeknek a Csökkentett Utasításkészletű Számítógépeknek. (CSUSZ. Azért ezt a rövidítést nem fogom használni . . .) Négy cikk foglalkozik a RISC gépek történetével, lényegével. Annak ellenére, hogy jelenleg a piacon nem sok ilyen sikeres gép található, a cikkek íróinak meggyőződése szerint a számítógépek további fejlesztése ebbe az irányba mutat.

A RISC gépek ősei az 1950-es és 60-as évek huzalozott logikájú gépei, valamint az 1971-ben született konvencionális mikroprocesszorok. Példaként a UNIVAC 1, az IBM 701, 704, az Intel 4004, 8008, 8080 és más gépek szerepelnek. Meglepő, hogy a Cray CDC 6600-as szupergépe a RISC filozófia legelső példája.

A mikroprogramozott processzorok az 1960-as évek közepén születtek, és az évtized végére a gépek cache-memóriával is bővültek. Ekkor a cél a minél összetettebb utasítások implementálása volt.

1985-re ismét a fix huzalozású processzorok (RISC) jelentek meg, két külön cache-memóriát alkalmazva az adatoknak és az utasításoknak. Ezek a gépek már széles körben használják a pipeline-technikát.

A RISC filozófia tudatos újbóli megjelenését a szerzők a technológia fejlődésével indokolják. A VLSI technológia sorra dönti a sebességi korlátokat, a tárműveletek egyre gyorsabbak. A MIPS mérőszám, a mikroprogramozott utasítások egységesedése következtében, a 70-es évek végére többé-kevésbé helyes összehasonlítást adott, de ez RISC gép esetén egyre kevesebbet mond számunkra.

Ezért a 70-es évek elején elkezdődött komplex teljesítményvizsgálati módszerek további fejlesztést igényelnek. Komplex rendszer-összehasonlító tesztekre van szükség, figyelembe véve az egyes alkalmazásokat, a hálózati környezetet, az üzemelő munkahelyek számát, azok rendszerre gyakorolt teljes terhelését.

BIT

(1986. május 16. Wehr, W.: Menschen, Möbel, Masse, Grundlagen der Ergonomie)

72 vagy 75 centiméter magas-e az íróasztal? A különbség lényegtelennek tűnik, hiszen mi az a 3 cm? Pedig igen sokat jelent.

A szerző a DIN 33402 szabvány második részéből kiindulva részletesen elemzi az irodabútorok méretezésének szempontjait. Milyen magas legyen a szék és az asztal? A karfa, ha van, hol legyen? Milyen távolságból tud az ember képernyős berendezéssel dolgozni? És így tovább.

A világítás a munkahely kialakításának fontos tényezője. Milyen erősségű világítás szükséges a különféle munkákhoz?

Igen lényeges, hogy az irodákban megfelelő legyen a mikroklíma. Maximum 26 °C környezeti hőmérséklet és 50—65 százalékos páratartalom engedhető meg. És a zaj? *Egyszerű* munkához 70 dB(A) zaj *még megengedhető.*

A környezet színe, fényviszszaverése, a munkahely tágassága, az íróasztal nagysága is mind fontos eleme a munkahely megfelelő kialakításának.

A cikk elég száraz. Felsorol, osztályoz, követelményeket ír le. Viszont azt hiszem, minden lényeges környezeti elemre kitér. Érdemes elolvasni, ha átalakítunk vagy újat építünk.

JANKÓ GÉZA

Távoktatás és informatika

Az „oktatás" szó hallatán a legtöbb embernek a padban ülő diák és a neki katedráról magyarázó tanár jut eszébe: a táblára felírt és a füzetbe átmásolt vázlatok és matematikapéldák. Vajon ha az oktatás elé írjuk a „táv" jelzőt, akkor képzeletünkben egymástól messze került, elszakadt katedrát és padsort látunk? Egy hiába beszélő tanárt és minden igyekezet ellenére semmit nem halló nebulókat? Nos, hogy közöttük kontaktus teremtődjék, valamiféle közvetítőeszköznek kell beiktatódnia. Tehát a távoktatás végül is az a tanítási forma, amikor a tanár és a diák közé egy bizonyos ismereteket hordozó ékelődik. Ezt az ismerettárat valamilyen technikai eljárással állítják elő, és a tanulni akaró személy önállóan, közvetlen oktatói segítség nélkül sajátítja el az eszköz közvetítette tananyagot.

Az elmondottak alapján távoktatási eszköznek minősíthetjük azokat a tankönyveket, amelyekben egy-egy anyagrész lényeges elemét, a begyakorlást elősegítő tesztek képezik, a nyelvtanulásban már elengedhetetlen kazetták, sőt meglepő módon egy-egy bonyolult használati tárgy kezelési utasítását is.

A közelmúltban bevezetett leglátványosabb példaként talán a televízió távoktatási adását, a TV BASIC-et említhetnénk.

A televíziós távoktatás esetében a hallgató számára a műsoridő még bizonyos kötöttséget jelent. Az utóbbi években egyre inkább előtérbe kerül, hogy a távoktatás a hallgató számára megfelelő időben foglalkozik az anyaggal, és maga ítéli meg, hogyan és hányszor ismételve sajátítja el azt.

Változnak az egyes szakmák, a szerzett tudás egyre gyorsabban elavul, az ismereteket állandóan korszerűsíteni kell. Nem a számítástechnikus szakmai elfogultsága, amikor ezt a tényt elsősorban a mikrogépek tömeges elterjedésével hozzuk párhuzamba.

Ezek a gépek munkaeszközzé váltak, használatukat annyi embernek kell megtanulnia, hogy ezek ki- és átképzése intézményesen megoldhatatlan. Egyrészt nincsen elég tanár és tanterem, másrészt a munkától elszakadva, iskolapadba beülni mind nehezebbé válik. Jól előkészített oktatási anyagokból, saját idejével gazdálkodva azonban a többség akar és tud időt szánni önmaga tovább- esetleg átképzésére.

Néhány országban már komoly eredményeket értek el a távoktatásban. Példaként említenénk a British Open Universityt, ahol mintegy 150 tantárgyból válogathatnak a hallgatók, s ha úgy tetszik, egyszerre tanulnak irodalomtörténetet és elméleti fizikát; vagy kuriózumként Thaiföldet, ahol a közép- és felsőoktatást túlnyomóan nyílt egyetem formájában oldják meg.

Mi lehet a leghatékonyabb eszköz és módszer? Nem kétséges, hogy a mikroszámítógép, amelyen az oktatóprogramok az ismeretek alapos besulykolására és „szigorú" kikérdezésére képesek.

Ily módon a számítógép a távoktatásban kettős szerepben jelenik meg: egyrészt mint az oktatás tárgya, másrészt mint az oktatás eszköze.

A Neumann társaság egyik legfontosabb feladatának azt tartja, hogy minél szélesebb körben terjessze a számítástechnikai ismereteket a felnőtt lakosság körében (hiszen a közép- és főiskolások oktatását intézményesen igyeknek megoldani). Ismerve a külföldi tapasztalatokat, szerettük volna, ha ezek Magyarországon az adott szakemberek előadásában hangzanak el, remélve, hogy az ügynek, az informatika távoktatásának hiveket szerzünk mind illetékesek, mind pedig lelkes végrehajtók személyében.

Az októberben megrendezett Teleteaching '86 konferencia igazolni látszik várakozásunkat. A húsz országból érkezett előadók és résztvevők sokféle eredményről, kísérletről, tervről számoltak be. A konferencia attrakciójának számító videodiszk-bemutató késő esti órákban is nagy érdeklődés mellett zajlott. A konferenciát Szabó Imre ipari minisztériumi államtitkár előadása nyitotta meg, amelyben — többek között — a magyarországi nyílt egyetem létrehozásának lehetőségéről is tájékoztatta a résztvevőket.

A távoktatás nemzetközi fontosságát mi sem igazolja jobban, mint hogy a résztvevők javaslatot terjesztettek az IFIP elé: alakítsa meg az oktatási bizottságán (TC 3) belül a felnőttoktatással foglalkozó, a kérdést jól ismerő szakembereket egyesítő munkacsoportját. Azt is javasolták, hogy a munkacsoport neve „distance learning" (távtanulás) legyen, ami egyben azt is jelzi, hogy az oktatásnak ez a módja az önálló tanulásra helyezi a hangsúlyt. Az ülés résztvevői a munkacsoport elnökének Kovács Győzőt, az NJSzT alelnökét, míg a következő, most már távtanulási konferencia — Teleteaching '87 — program- és szervezőbizottsága elnökének dr. Wichit Srisa-an-t, a thaiföldi nyílt egyetem rektorát választották. A konferenciára 1987 végén kerül sor Bangkokban.

TÓTH ISTVÁNNÉ

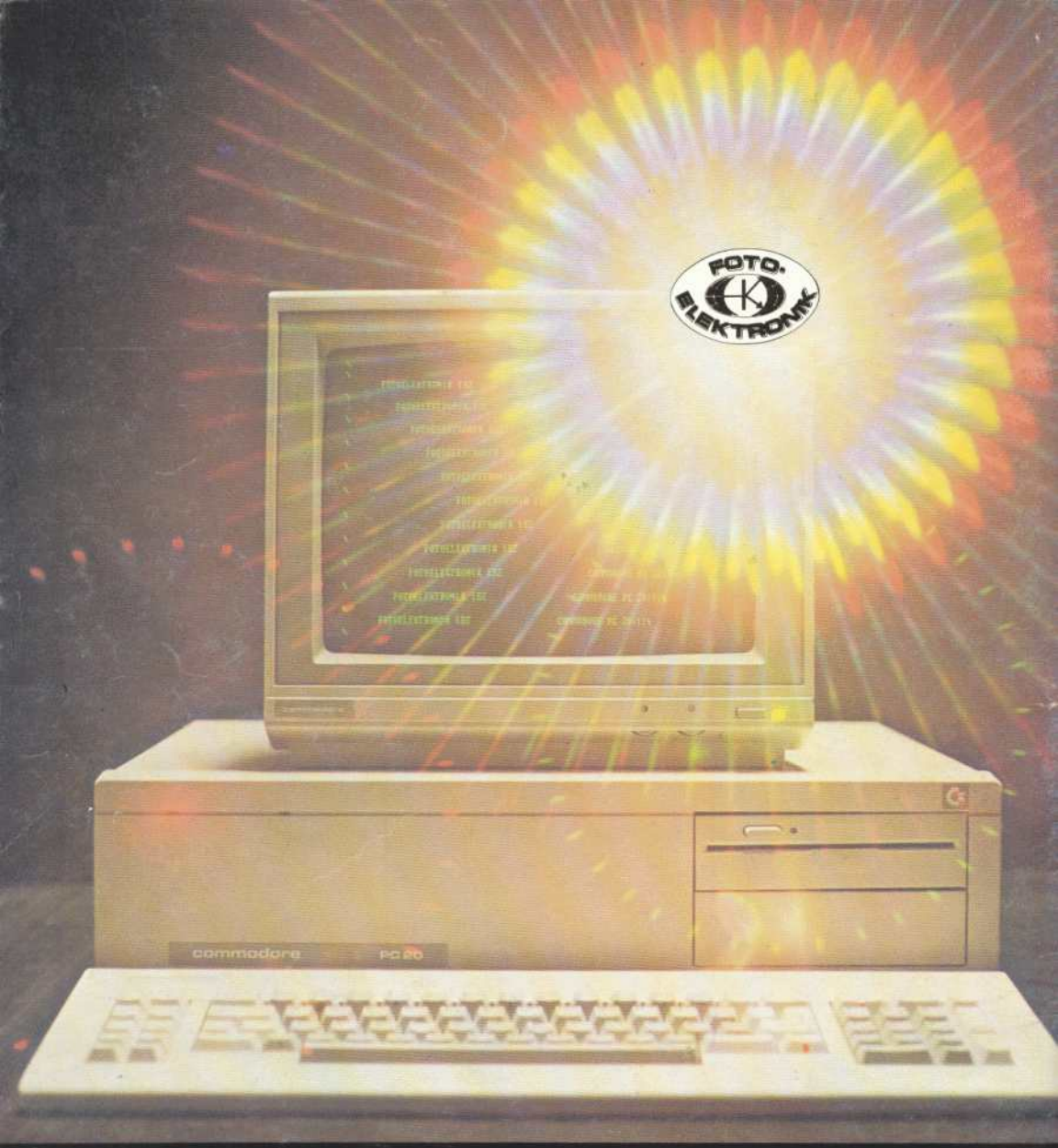
Sylvia Charp (USA) a Nemzetközi Információfeldolgozási Szövetség (IFIP) Oktatási Bizottságának tagja az egyesült államokbeli tapasztalatokról számol be Poór Klárinak

A kiállítók között örvendetes számban szerepeltek magyarországi oktatási intézmények

Szeretettel várjuk nagy karácsonyi vásárunkra!

Budapest XIII. Balzac u. 35. Telefon: 402-954

Kellemes karácsonyi ünnepet kívánunk minden kedves vásárlónknak!



MIRE VAN SZÜKSÉGE? SZÁMÍTÓGÉPRE? VIDEORA? KÜLÖNLEGES MÜSZAKI CIKKEKRE?

ÉVVÉGI BEVÁSÁRLÁSAIBAN ISMÉT SEGÍT A FOTOELEKTRONIK